HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BACHELORARBEIT

Herr

Imre Katona

Weiterentwicklung eines Programms zur Analyse und Visualisierung von Genexpressionsdaten

Mittweida, 2013

Fakultät
Mathematik/Naturwissenschaften/Informatik

BACHELORARBEIT

Weiterentwicklung eines Programms zur Analyse und Visualisierung von Genexpressionsdaten

Autor:

Herr

Imre Katona

Studiengang:

Biotechnologie/Bioinformatik (B. Sc.)

Seminargruppe:

BI10w1-B

Erstprüfer:

Prof. Dr. rer. nat. Dirk Labudde

Zweitprüfer:

Dr. Alexander Herr

Einreichung:

Mittweida, 23. September 2013

Bibliografische Beschreibung:

Katona, Imre: Weiterentwicklung eines Programms zur Analyse und Visualisierung von Genexpressionsdaten, 64 Seiten, 28 Abbildungen, 3 Tabellen, Software DVD, Hochschule Mittweida (FH), MNI

Bachelorarbeit, 2013

Englischer Titel:

further development of a program for analysis and visualization of gene expression data

Kurzbeschreibung:

Für die Analyse und Visualisierung von Genexpressionsdaten existiert ein Programm der Firma Biotype® Diagnostic GmbH: der XpressionXplorer. Dieser liest normalisierte Genexpressionsdaten des "GeneChip® Human Genome U133 Plus 2.0 Array" von Affymetrix ein, die von den Datenbanken GEO und ArrayExpress abstammen. Die Genexpressionsdaten stellen eine zentrale Datenstruktur dar, welche sich abstrahiert als Tabelle, mit einer fixen Anzahl von 54.675 Zeilen und einer variablen Anzahl von über 9.000 Spalten, vorstellen lässt. Jede Zeile der Tabelle repräsentiert ein Probeset auf dem GeneChip® und jede Spalte ein Genexpressionsprofil aus einem Microarray-Experiment. Die vorliegende Arbeit beschäftigt sich mit der Weiterentwicklung einer Programmkomponente des XpressionXplorers. Die Programmkomponente hat die Bezeichnung 'Differential Expression' und dient zur Entdeckung von Genen, die sich als Biomarker, z. B. von Tumorerkrankungen, eignen. Diese Programmkomponente sollte durch Einbinden einer neuen grafischen Komponente - der VirtualTreeview - erweitert und verbessert werden. Die hauptsächlichen Vorteile dieser VirtualTreeview sind die kurzen Aktualisierungszeiten, die zusätzliche Darstellung in Spalten und die Möglichkeit des interaktiven Bedienens durch die Benutzer.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

bp	Basenpaare (base pairs)
cDNA	complementary DNA
Cy3	Cyanine 3
Cy5	Cyanine 5
dbEST	Expressed Sequence Tags database
DNA	Desoxyribonukleinsäure (deoxyribonucleic acid)
dT	Desoxythymidin
EBI	European Bioinformatics Institute
ESTs	Expressed Sequence Tags
FPC	Free Pascal Compiler
GC	Guanin-Cytosin
GEO	Gene Expression Omnibus
GUI	Graphical User Interface
HUGO	Human Genome Project
IDE	Integrated Development Environment
kb	Kilobasenpaar
LCL	Lazarus Component Library
MAGE-ML	Microarray Gene Expression Markup Language
MIAME	Minimum Information about a Microarray Experiment
mRNA	messenger RNA
NCBI	National Center of Biotechnology Information
nt	Nucleotide
PAP	Programmablaufplan
PCR	Polymerase-Chain-Reaction
RAM	Random-Access Memory

RefSeq	Reference Sequence database
RNA	Ribonucleinsäure (Ribo-Nucleic-Acid)
rRNA	ribosomal RNA
SAGE	Serial Analysis of Gene Expression
SMD	Small-Molecule-Drugs
SNP	Single Nucleotide Polymorphism
snRNA	small nuclear RNA
ssDNA	Einzelstrang (single strand) DNA
STRs	Short Tandem Repeats
STSs	Sequence Tagged Sites
T	Thymin
tRNA	transfer RNA
VCL	Visual Component Library
VST	Virtual String Tree
XML	Extensible Markup Language

Danksagung

„Man muss ins Gelingen verliebt sein, nicht ins Scheitern.“

- Ernst Bloch (*1885, †1977)

An dieser Stelle möchte all den Menschen danken, die immer an meiner Seite sind und besonders denjenigen, die mich bei meiner Bachelorarbeit unterstützt haben.

In erster Linie danke ich Professor Dr. rer. nat. Dirk Labudde dafür, dass er sich als Betreuer seitens der Hochschule zur Verfügung gestellt hat und in mir den Eifer für das Thema dieser Bachelorarbeit geweckt hat. Ebenso gilt mein Dank meinem Betreuer auf Unternehmensseite, Dr. Alexander Herr, für seine freundliche Unterstützung, sowie für seine lehrreichen Erklärungen, welche mir bei meiner Arbeit überaus hilfreich waren.

Weiterhin möchte ich mich bei meiner Freundin Jessica Mai und bei meinen Kommilitonen Christoph Leberecht und Eric Zuchantke für das Korrekturlesen und die Hilfe beim Programmieren bedanken. Besonders danke ich Jessica dafür, dass sie auch in schwierigen Momenten immer einen Weg gefunden hat mich aufzubauen.

Last but not least möchte ich mich recht herzlich bei meinen Eltern dafür bedanken, dass sie mich während meines Studiums des Öfteren moralisch, aber auch finanziell, sehr unterstützt haben.

1 Einleitung

Erst als die vollständige Entschlüsselung des humanen Genoms im Rahmen des Humangenomprojekts (HUGO) gelang, und dessen Veröffentlichung 2001 erfolgte, wurde es möglich unterschiedliche Phänotypen, sowie auch Mutationen die zu Krankheiten führen, auf ihre genetischen Ursachen hin zu untersuchen. Nach und nach konnten dadurch auch neue Regulationsmechanismen, welche die Genexpression beeinflussen, entdeckt und analysiert werden. Denn jede menschliche Zelle (außer Spermien und Eizellen) besitzt einen kompletten Satz der rund 23.000 Gene und somit ein vollständiges Genom. Aber das Produkt der Genexpression, das als Proteom bezeichnet wird, ist abhängig von den Proteinen, die in einer Zelle benötigt werden. Dies ist wiederum bedingt durch die Differenzierung der jeweiligen Zelle und durch bestimmte Umweltfaktoren wie z. B. Temperatur, pH-Wert und der benachbarten Zellen, die die jeweilige Zelle umgeben. Somit wird nicht jedes Gen in der Zelle transkribiert und exprimiert. Herauszufinden welche Gene in einer bestimmten Zelle oder einem Zellverband exprimiert werden ist das Ziel der Genexpressionsanalyse. Zu den Methoden der funktionellen Analyse des Genoms zählen DNA-Microarrays, welche sich als Hochdurchsatzverfahren hervorragend zur Bestimmung der zellulären Genexpression eignen. Durch diese Hochdurchsatzverfahren werden enorm viele Daten erzeugt, was einen hohen Anspruch an die Bioinformatik stellt, diese zu verwalten und auszuwerten [SELZER (2004), S. 137f].

In diesem Kapitel werden die Anfänge der Sequenzierung in den 1990er Jahren bis hin zu den heutigen Analysemethoden mittels DNA-Microarray-Technologien beschrieben. Speziell für die Sequenzierung des humanen Genoms sowie anderer eukaryotischer Genome war und ist es nötig im Vorfeld spezifische Marker zu gewinnen, die mittels Klonierungstechnik aus der gesamten speziesspezifischen mRNA generiert werden. Diese spezifischen Marker werden zu Beginn dieses Kapitels im Abschnitt 1.2 erläutert. Weiterhin wird auf die unterschiedlichen Mechanismen zur Regulierung der Genexpression bei Eukaryoten eingegangen, die im Laufe der Jahre entdeckt wurden. Die zelluläre Genexpression kann mittels der modernen DNA-Microarray-Technologien untersucht werden. Die dadurch generierten Datenmengen sind gigantisch und müssen deshalb durch bioinformatische Techniken aufgearbeitet werden, um brauchbare Ergebnisse zu bekommen. Die Bioinformatik ist ein interdisziplinärer

Wissenschaftszweig, der sich seit der Sequenzierung des humanen Genoms 2001 gebildet hat, und seither immer mehr an Bedeutung gewinnt. Zur Speicherung und Verwaltung der mittels DNA-Microarrays gewonnenen Daten werden bioinformatischen Datenbanken vorgestellt, in denen die aufgearbeiteten Datensätze aus Microarray-Experimenten zusammengetragen werden und zur weiteren Analyse zur freien Verfügung stehen.

1.1 Motivation

Die Genexpressionsanalyse ist der nächste logische Schritt, den die Sequenzierung von Genomen nach sich zieht. Aufgabe der Genexpression ist es mittels geeigneter Hochdurchsatz-Methoden wie bspw. DNA-Chips zu ermitteln, wie stark spezielle Gene transkribiert werden. Dies geschieht indirekt durch die Bestimmung der mRNA-Konzentration in den entsprechend dazu aufgearbeiteten Proben. Diese können je nach Aufgabenstellung aus einem beliebigen Pro- oder Eukaryoten stammen, dessen Genom bereits sequenziert ist, ebenso wie das des *Homo sapiens*. Die Analyse von Genexpressionsdaten hat die Klärung mehrerer wissenschaftlicher und medizinischer Fragen zum Ziel [MADIGAN (2006), S. 564; MÜLLER (2004), S. 8]:

- Wie unterscheidet sich gesundes von entartetem Gewebe und wie lässt sich dies differenzieren (bspw. durch das Finden von Biomarkern)?
- Wie ändert sich das Transkriptionsmuster im Laufe des Lebens eines Patienten oder eines Organismus?
- Was zeichnet ein bestimmtes Organ auf Transkriptionsebene aus?
- Wie kann man die Komplexität von Stoffwechselprozessen und deren physiologischen Regulationsmechanismen verstehen?
- Wie beeinflusst der Biorhythmus die Expression (Also die Frage nach inneren Uhren)?
- Welchen Einfluss haben Umweltgifte oder Pharmaka auf die Expression?
- Worauf beruhen Erkrankungen, wie Fettleibigkeit, Arteriosklerose oder Asthma?

Zur Beantwortung dieser Fragen rückten im Laufe des letzten Jahrzehnts Teilgebiete wie die Systembiologie, die Pharmakogenetik und die Pharmakogenomik immer mehr in den Fokus der biologischen und medizinischen Forschung. In der Tat entwickelte sich aus den Anwendungen der molekularen Biologie auch eine neue Art der Diagnose und Therapie: die molekulare Medizin [HOFFMANN (2006), S. 5].

Die molekulare Medizin konzentriert sich im Wesentlichen auf die Untersuchung molekularer Grundlagen der Pathogenese und untersucht, ob sich daraus möglicherweise Ansätze zur pharmakologischen Intervention ergeben [URL-1]. Einige praktische Beispiele, die aus diesem Forschungszweig hervorgehen, sind:

- Das Brustkrebsmittel Herceptin, was nur bei Patienten wirkt, deren Körper den HER2-Rezeptor für eine Tyrosinkinase überexprimiert. Einer pharmakologischen Therapie mit Herceptin müssen daher molekular-medizinische Tests vorrausgehen [HOFFMANN (2006), S. 20f].
- Die Enzymgruppe Cytochrom P450 ist bei einigen Menschen sehr aktiv und bei anderen dagegen kaum. Dadurch ergibt sich eine variable Stoffwechselrate für den Abbau von Medikamenten, die auch mit teils schwerwiegenden Nebenwirkungen einhergehen kann [HOFFMANN (2006), S. 24f].

1.2 Die Sequenzierung kompletter Genome

Im Jahre 1995 wurde durch die erste vollständige Sequenzierung eines prokaryotischen Genoms, das des Grippeerregers *Haemophilus influenzae*, ein Meilenstein in der Genomforschung gelegt. Somit wurde es erstmals möglich die Gene dieses Genoms zu analysieren und auf ihre regulatorischen Prozesse hin zu untersuchen. Drei Jahre später, im Jahre 1998, war die vollständige Sequenzierung des ersten eukaryotischen Genoms, das des Fadenwurms *Caenorhabditis elegans*, vollbracht. Die eukaryotischen Genome sind viel größer als die der Prokaryoten und besitzen komplexere Regulationsmechanismen, wie bspw. den Vorgang des Spleißens. Während bei Prokaryoten fast das gesamte Genom aus proteincodierenden Genen besteht, so stellen bei Eukaryoten die Gene zur Synthese von Proteinen nur einen Bruchteil des gesamten Genoms dar [SELZER (2004), S. 91].

So besteht bspw. das menschliche Genom zu nur rund 5 % aus codierenden DNA-Bereichen. Die Größe des gesamten Genoms beträgt in etwa 3,27 x 10^9 Basenpaare und beinhaltet ca. 23.000 Gene (die Angaben hierzu variieren je nach Literaturstelle zwischen 20.000 bis 30.000), womit sich eine durchschnittliche Gendichte von 10 pro 1 Millionen Basenpaaren ergibt. Die nicht-codierenden DNA-Abschnitte können wichtige regulatorische Abschnitte wie bspw. Bindungsstellen für Transkriptionsfaktoren sein. Daneben können sie für nicht proteincodierenden Gene von Bedeutung sein oder bei der Bildung der Chromosomenstruktur eine wichtige Rolle spielen [URL-2].

1.2.1 Charakterisierung von Genomen mit STSs

Die Entschlüsselung des menschlichen Genoms erforderte enorme Anstrengungen, da über drei Milliarden Nukleotide sequenziert und in silico zusammengesetzt werden mussten. Um dieses Projekt zu realisieren wurden bestimmte Orientierungspunkte (Marker) festgelegt, die als "Sequence Tagged Sites" (STSs) bezeichnet werden. STSs sind DNA-Sequenzen mit einer Länge von 200 bis 500 nt und kommen nur einmal im Genom vor, womit sie sich sehr gut als Marker für die Kartierung von Chromosomen bzw. Genomen eignen. Generiert werden STSs durch die Amplifikation spezifischer Nukleotidsequenzen mit Hilfe der PCR-Methode. Diese Amplikons werden in Vektoren kloniert, um eine DNA-Bibliothek zu erzeugen. Die entstandenen DNA-Klone können infolgedessen durch Datenbankabfragen auf die Existenz von passenden STS-Bereichen hin untersucht werden. Mittels dieser Informationen können die DNA-Klone in der richtigen Position auf den Chromosomen des Genoms platziert werden. Mit Hilfe dieser Technik konnte eine präzise physikalische Karte des humanen Genoms erstellt werden [SELZER (2004), S. 92f]. Bereits 1989 erkannten Wissenschaftler, dass STSs auch von cDNA-Klonen erstellt werden können. Da cDNA-Klone von der mRNA abstammen, kodiert ihre Sequenzabfolge eine Aminosäuresequenz und somit ein Protein. Dadurch ist es möglich STSs zur Lokalisierung von Genen im Genom einzusetzen. Auf diese Weise wurde im Jahre 1996 auch die Erzeugung einer Genkarte des menschlichen Genoms, im Rahmen des HUGO, realisiert [SELZER (2004), S. 93].

1.2.2 Charakterisierung von Genomen mit ESTs

Als "Expressed Sequence Tags" (ESTs) werden Teilsequenzen von cDNA-Klonen bezeichnet, welche hervorragend zur Entdeckung neuer Gene geeignet sind. Grundlage dafür ist, dass cDNA-Klone von mRNA und somit von exprimierten Genen abstammen. Aufgrund der Tatsache, dass die Generierung von ESTs Mitte der 1990er Jahre immer mehr mittels Hochdurchsatzverfahren durchgeführt werden konnten, entstand ein regelrechter Boom von EST-Projekten. Als Ansatz jener Projekte diente ein spezifisches Ausgangsmaterial (Zellen, spezifische Gewebe oder im Einzelfall ganze Organismen) zur Erzeugung einer cDNA-Bibliothek. Aus diesem Ausgangsmaterial wird die Gesamt-RNA isoliert. RNA-Moleküle sind immer einzelsträngig, aber können Doppelstrangbereiche ausbilden und so funktionelle, bzw. katalytische Strukturen bilden (wie z. B. die Kleeblattstruktur der tRNA). Interessanterweise enthalten Zellen wesentlich mehr RNA als DNA. Man unterscheidet die RNA prinzipiell in codierende und nicht-codierende. Zu den nicht-codierenden zählen bspw. die tRNA, rRNA, snRNA und

weitere. Eine Klassifizierung der RNA ist in Tabelle 1 dargestellt [LÖFFLER (2008), S. 244f]. Nur aus der codierenden mRNA, die lediglich einen Anteil von ca. 2 bis 3 % an der Gesamt-RNA ausmacht, lassen sich ESTs generieren. Um die labile mRNA zu stabilisieren, wird sie deswegen zuerst mittels Reverse-Transkriptase in cDNA umgeschrieben. Diese cDNA wird dann oft gerichtet, so dass das 3'- und 5'-Ende bekannt ist, in Plasmide kloniert. Diese Plasmide werden danach in *Escherichia coli*-Bakterien transformiert und in diesen repliziert. Auf diese Weise entsteht eine cDNA-Bibliothek. Aus dieser werden nach Bedarf *Escherichia coli* Bakterien auf Nährmedien ausplattiert und selektiert, um aus ihnen Plasmid-DNA zu isolieren. Diese cDNA wird anschließend sequenziert und die Nukleotidsequenzen in Form von Rohdaten in einen Computer exportiert. Dort werden sie schließlich in silico bioinformatisch aufbereitet [SELZER (2004), S. 93-98].

Tabelle 1 - Klassifizierung der RNA [bearbeitet nach LÖFFLER (2008), S. 245]

Typ	Bezeichnung	Funktion
Codierende RNA	Messenger RNA (mRNA)	Matrize bei der Proteinbiosynthese
Nicht codierende RNA (ncRNA) Strukturell katalytische Funktion	Transfer RNA (tRNA)	Translation der genetischen Information
	Ribosomale RNA (rRNA)	Strukturelement der Ribosomen Katalysator bei der Knüpfung der Peptidbindung
	Small nuclear RNA (snRNA)	Spleißen der Prä-mRNA: Strukturelement der Spleißosomen
	Small nucleolar RNA (snoRNA)	Modifikation von RNA (z. B. Methylierung von Riboseresten)
	Signal Recognition Particle-RNA (SRP)	Intrazellularer Proteintransport
	Ribonuclease P-RNA	Reifung der Prä-tRNA
	Telomerase-RNA	DNA-Synthese an den Telomeren
Nicht codierende RNA (ncRNA) Regulatorische Funktion	Mikro-RNA (miRNA) Small interfering RNA (siRNA)	Abbau von mRNA
	Xist-RNA	Inaktivierung des X-Chromosoms

1.3 Genetische Ursachen für individuelle Phänotypen

Das Genom von eukaryotischen Zellen ist durch das Vorhandensein von Mutationen geprägt. Diese Mutationen stellen die genetische Vielfalt dar und sind deshalb auch als Variationen zu betrachten. Zu diesen Variationen zählen u.a. die Punktmutationen, auch bekannt unter dem Fachbegriff "Single Nucleotide Polymorphisms" (SNPs). Diese werden durch den Austausch eines einzelnen Nukleotids auf dem Matrizenstrang und dem dazugehörigen Komplement hervorgerufen. Neben den SNPs treten weitere Polymorphismen auf. Zu diesen gehören die Deletion und Insertion einzelner Nukleotide

oder kürzerer Nukleotidabschnitte. Weiterhin sind seit längerem sogenannte "Short Tandem Repeats" (STRs) bekannt [SELZER (2004), S. 106]. STRs sind kurze sich hintereinander wiederholende bp-Muster in einem RNA- oder DNA-Strang [URL-17]. Diese repetitiven Sequenzabschnitte werden in manchen Zusammenhängen auch als Mikrosatelliten bezeichnet. Die STRs treten meist in nichtkodierenden Bereichen der DNA auf und sind zur Zeit die gebräuchlichsten DNA-Motive, die für die genetische Individualisierung von Personen verwendet werden [URL-17]. Sie spielen daher bei der Erzeugung von DNA-Profilen eine entscheidende Rolle. In der Forensik ist, für das Erstellen solcher DNA-Profile von Personen, der Begriff des genetischen Fingerabdrucks gebräuchlich.

Von den zuvor beschriebenen Mutationen sind die SNPs für die Wissenschaft von besonderem Interesse, da sie im menschlichen Genom die Mutationen sind, die am häufigsten auftreten. Aus diesem Grund hat sich ein Konsortium aus kommerziellen und nicht kommerziellen Mitgliedern gebildet, welches es sich zur Aufgabe gemacht hat möglichst viele dieser SNPs im humanen Genom zu identifizieren. Im November 2012 veröffentlichte dieses Konsortium eine Studie, in der 1.092 Individuen aus 14 verschiedenen Populationen auf Polymorphismen hin untersucht wurden. Als Ergebnis lieferte diese Studie in der Summe circa 38 Millionen SNPs, 1,4 Mio. kurze Insertionen und Deletionen, sowie über 14.000 längere Deletionen. Mittlerweile ist also eine riesige Anzahl von Polymorphismen im humanen Genom identifiziert worden und es werden vermutlich, im Verlauf der nächsten Jahre, noch mehr dazukommen [URL-3].

Bei der Identifizierung von SNPs ist es wichtig zu unterscheiden, ob sie sich in codierenden oder in nicht-codierenden Bereichen der DNA befinden. Denn auf nicht-codierenden Bereichen, außerhalb von Genen, haben die SNPs keinen Einfluss auf die Struktur oder Funktion der Zellproteine. Daher besteht in der Genomik ein besonderes Interesse darin, speziell die SNPs herauszufinden, welche auf den codierenden Bereichen der DNA-Moleküle liegen. Da der Informationsfluss immer vom Genom zum Proteom verläuft und nicht umgekehrt, lassen sich durch die gesammelten Erkenntnisse zu den SNPs in den codierenden DNA-Bereichen, Zusammenhäng zu den Phänotypen herstellen. Phänotypen beim Menschen sind z. B. die Augen- und Haarfarbe, aber können ebenso auch Erbkrankheiten sein. Stehen bestimmte SNPs in Zusammenhang mit Phänotypen, so werden sie als funktionell bezeichnet. Um funktionelle SNPs zu identifizieren sind Studien mit zufällig ausgewählten Individuen notwendig, die die Häufigkeit eines speziellen SNPs mit dem Auftreten eines Phänotyps vergleichen, um eine mögliche Korrelation herstellen zu können [SELZER (2004), S. 106f].

1.4 Regulation der Genexpression

Ein großer evolutionsbiologischer Vorteil von Zellen, ist ihre Fähigkeit die Genexpression zu regulieren um somit auf variable Umweltbedingungen zu reagieren. Die Vielfalt der Regulationsmechanismen bei eukaryotischen Zellen ist dabei beachtlich. So können bspw. Gene durch Methylierung inaktiviert oder der Abbau von mRNA verstärkt oder vermindert werden. Die Methylierung erfolgt in den Promotorregionen der Gene, wobei im Speziellen Cytosinreste spezifischer GC-Paare methyliert werden, wodurch die Expression dieser Gene unterdrückt wird. Eine weitere Regulation kann durch Veränderungen an den Histonen, durch Acetylierung, Phosphorylierung und Methylierung, erfolgen. Die Phosphorylierung dieser Histone erfolgt u.a. durch Proteinkinasen, welche durch extrazelluläre Signalmoleküle, wie z. B. Hormone, aktiviert werden. Während die Histonacetylierung und –phosphorylierung reversibel sind, so wird von der Histonmethylierung angenommen, dass sie irreversibel ist und damit nur für die Inaktivierung von Genen bedeutsam ist. Weiterhin kann die Transkription von Genen durch Enhancer um ein Vielfaches stimuliert werden. Enhancer sind Proteine, die durch die Bindung von Liganden, wie z. B. Steroidhormonen und Xenobiotika, aktiviert werden und damit als Transkriptionsaktivatoren fungieren. Eine Übersicht aller zuvor beschriebenen, sowie weiterer Mechanismen zur Regulierung der Transkription, und damit korrelierender Genexpression, ist in Tabelle 2 dargestellt [LÖFFLER (2008), S. 257-259].

Tabelle 2 – Übersicht der Mechanismen zur Regulierung der Transkription [bearbeitet nach LÖFFLER (2008), S.257]

Regulierter Schritt	Mechanismus	Vorkommen (Beispiele)
(In-)Aktivierung von Genen	Inaktivierung durch Methylierung an GC-Paaren, Aktivierung durch Demethylierung	Viele von der Zelldifferenzierung abhängige Gene
Veränderungen der Chromatinstruktur (u.a. Histone)	Reversible Acetylierung und Phosphorylierung von Histonproteinen	Viele durch extrazelluläre Signalmoleküle regulierbare Gene
Initiation der Transkription	Aktivierung des Transkriptions-komplexes durch ligandenaktivierte Transkriptionsfaktoren	Aktivierung der Trans-kription durch Steroid-hormonrezeptoren, u.a.
Alternatives Spleißen	Verwendung alternativer Spleißstellen	Isoformen des D2-Dopaminrezeptors
RNA-editing	Posttranskriptionale Modifikation von Basen auf der mRNA	Erzeugung von Isoformen des Apolipoprotein B
Abbau der RNA	Verhinderung des endonucleo-lytischen Abbaus durch RNA-Bindungsproteine	Stabilität der Transferrin-rezeptor-mRNA
	RNA-Interferenz	Abbau spezifischer mRNA-Moleküle, Abwehr von RNA-Viren, Regulation der Zelldifferenzierung

Wie in Tabelle 2 erkennbar ist, wird die Genexpression häufig auf der Ebene der Transkription reguliert. Wenn man die Bedingungen untersucht, unter denen ein bestimmtes Gen transkribiert wird, so lassen sich daraus auch Rückschlüsse auf die Funktion der jeweiligen Gene ableiten. Um das Transkriptom zu untersuchen, wird die gesamte RNA, die zu einem bestimmten Zeitpunkt in einer Zelle vorliegt, analysiert [MADIGAN (2006), S. 562]. Für diese Analyse haben sich DNA-Microarrays als die Hochdurchsatzverfahren, die sich am besten zur Bestimmung der zellulären Genexpression eignen, durchgesetzt [SELZER (2004), S. 138].

1.5 DNA-Microarrays zum Durchführen von "Expression-Profiling"-Experimenten

Mittels Microarrays ist es möglich, von jeder Zelle anhand der exprimierten Gene ein Profil zu erstellen. Daher bezeichnet man die Methode auch als "Expression-Profiling". Als Trägermaterial für DNA-Microarrays dient oft eine Glasplatte, die in etwa die Größe eines Objektträgers aus der Mikroskopie besitzt. Auf dieser Glasplatte sind mehrere tausend DNA-Spots aufgebracht. Ein DNA-Spot ist eine Sammlung von vielen Kopien einer einzelsträngigen DNA, die so einzigartig ist, dass dadurch eine eindeutige Zuordnung zu einem spezifischen Gen ermöglicht wird [SELZER (2004), S. 139].

DNA-Microarrays werden je nach eingesetztem ssDNA-Typ der Spots unterschieden in cDNA-Arrays und Oligonukleotid-Arrays [SELZER (2004), S. 139]. Für cDNA-Arrays werden meist PCR-amplifizierte cDNA-Fragmente mit einer Länge von 0,6 bis 2,4 kb gespottet. Üblich sind cDNA-Arrays mit bis zu 10.000 Spots, jedoch kann auch eine Größe von bis zu 30.000 cDNA-Spots erreicht werden [URL-6]. Die cDNA-Arrays sind aufgrund ihrer langen ssDNA-Fragmente hoch spezifisch, aber aufwendig in der Herstellung. Denn diese erfordert das Anlegen einer cDNA-Bibliothek mit tausenden verschiedenen cDNA-Klonen im Labor. Im Gegensatz dazu sind Oligonukleotid-Arrays mittels photolithographischer Festphasensynthese, die ursprünglich aus der Halbleiterfertigung stammt, sehr schnell produzierbar. In Abbildung 1 wird der Herstellungsprozess schematisch dargestellt. Das Verfahren der photolithografischen Festphasensynthese hat den Vorteil, dass damit eine sehr hohe Dichte der aufzutragenden Spots (250.000 pro cm^2) zu erreichen ist [MÜLLER (2004), S. 71]. Dafür ist aber im Vorfeld das sorgfältige Design der synthetisierten Oligonukleotide (Oligos) von essentieller Bedeutung, die dann auf dem Objektträger erzeugt werden. Nur mit den richtigen Oligos lässt sich der Erfolg bei den nachgestellten Microarray-Experimenten garantieren. Diese Oligos sind 20 bis 50 nt lang und werden aus verschiedenen Datenbanken wie der GenBank®, dbEST und RefSeq abgeleitet [URL-4].

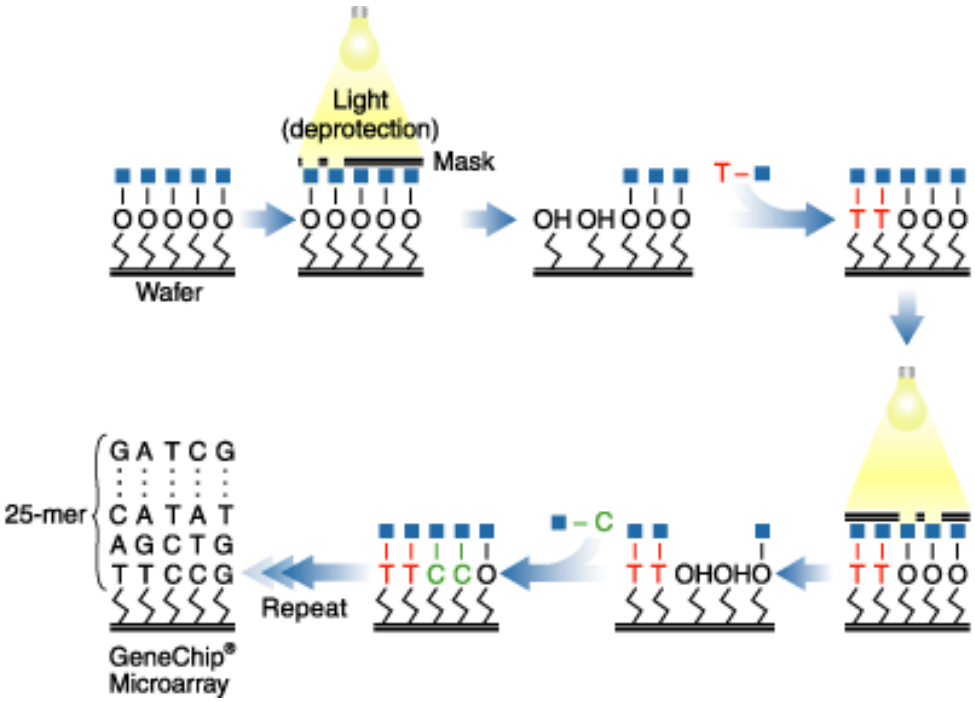


Abbildung 1 - Schema zur photolithographischen Herstellung von Oligonukleotid-Arrays [URL-5; Bildunterschrift von URL-6]
Silanisierte Quarzobjektträger mit gekoppelten photolabilen Linkermolekülen, die durch Licht aktiviert werden, werden mit entsprechenden Masken partiell abgedeckt. Durch Lichteinfall werden die Schutzgruppen der Linker entfernt. Eine Lösung mit einem Typ Didesoxynukleotiden (ddT), an die ebenfalls eine Schutzgruppe gebunden ist, wird über die Oberfläche gegeben, woraufhin sich die Nukleotide an die aktivierten Linker binden können. Im nächsten Syntheseschritt wird eine andere Maske über den Objektträger gelegt und der Vorgang wird wiederholt.

Ein Beispiel für einen Oligonukleotid-Array ist der "GeneChip® Human Genome U133 Plus 2.0 Array" von Affymetrix. Dieser ermöglicht die Analyse des gesamten menschlichen Transkriptoms auf einem einzigen Microarray. Zum Transkriptom zählen alle RNA-Moleküle, die zu einem bestimmten Zeitpunkt in einer Zelle existiert. Auf dem GeneChip® sind genau 54.675 Probesets aufgebracht. Die Probesets stellen ein Cluster von Sequenzen dar, und wurden mithilfe der UniGene Datenbank kreiert. Die Probesets bestehen meistens aus jeweils 11 Probes und jedem Probeset ist eine eindeutige Identifikationsnummer zugeordnet. Jedes Probe besteht aus einem 25nt-langem Oligonukleotid, an dem das komplementäre Transkript hybridisierten kann. In Abbildung 2 ist ein Probe mit seinem entsprechendem Transkript dargestellt. Mittels eines GeneChips® lässt sich der Expressionsgrad von über 47.000 Transkripten (RNAs) und deren Varianten testen. 38.500 dieser Transkripte sind humane Gene, die durch vorangegangene Forschungsarbeiten bereits gut beschrieben wurden [URL-13].

Probe info					Probe alignment
text	X	Y	inter pos	sts	alignment
200034_s_at1	831	531	367		chr18 complement(6452657..6452681) Probe g c a a a a t g c c t a g a t a t t a t c c t a c Genome g c a a a a t g c c t a g a t a g t a t c c t a c SNP Status = **'MM'** temperature=**undefined**

Abbildung 2 - Beispiel eines 'Probe' [URL-15]
Abgebildet ist eines von elf Probes aus einem Probeset mit der Bezeichnung '200034_s_at'. Das Probe besteht aus einem 25 nt-langem Oligo. Der komplementäre Strang des Oligos ist ein Transkript des Gens 'LOC645387' - lokalisiert auf dem Chromosom 18 des humanen Genoms.

Die meisten "Expression-Profiling"-Studien haben den Vergleich unterschiedlicher Genexpressionsmuster zum Ziel. So z. B. den Vergleich des Genexpressionsprofils einer gesunden Zelle (Kontrolle) mit dem einer Tumorzelle (Probe). Um beide Proben gleichzeitig untersuchen zu können bieten sich cDNA-Arrays an. Um die Proben auf den cDNA-Array aufzubringen, muss vorab die Gesamt-RNA aus beiden Zelltypen aufgereinigt werden, da die in dem Gemisch vorhandene mRNA nur ca. 2 bis 3 % der Gesamt-RNA ausmacht. Die bisher beste Methode, um mRNA aus Gesamt-RNA zu isolieren, ist die Selektion durch Poly-Adenylierung der mRNA. Die Poly-A^+-RNA bindet dann an Oligo-dT-Nucleotide, die eine Länge von ca. 20 nt besitzen und an magnetischen Kügelchen, Cellulose- oder Latexkügelchen gebunden sind. Nach der Aufreinigung hat man den Anteil der mRNA auf ca. 50 % gesteigert. Für einen höheren Anteil muss die Gesamt-RNA zweimal gereinigt werden. Die mRNA kann nun in cDNA umgeschrieben werden. Das Prinzip funktioniert so, dass man die immobilisierte mRNA mit Primern (z. B. Poly-T-Primern) zusammen pipettiert und den Ansatz auf ca. 70 °C erhitzt, um die Oligo-dT-"Angeln" mit der hybridisierten mRNA aufzuschmelzen. Nach Abkühlung des Ansatzes auf Raumtemperatur, können die Primer mit der mRNA hybridisieren und mit Hilfe von Reverse-Transkriptase verlängert werden (Primer-Extension). Folglich wird die cDNA von jedem Zelltyp mit einem anderen Fluoreszenzfarbstoff markiert, so dass sie später bei der Detektion unterschieden werden können. Die Kontrolle wird meist mit dem Farbstoff Cyanine 3 (Cy3) versehen, welcher nach Anregung grünes Licht emittiert. Die Probe hingegen wird oft mit dem Farbstoff Cy5 markiert, der bei Anregung rotes Licht emittiert. Danach werden Kontrolle und Probe auf den cDNA-Array pipettiert. Die mit Fluoreszenzfarbstoff markierten cDNA-Abschnitte hybridisieren nun mit den Oligos auf dem Träger. Die Menge der gebundenen cDNA an den Spots des cDNA-Arrays lassen sich schließlich quantifizieren. Dazu wird ein Laser eingesetzt, der mit den entsprechenden

Emissionsfrequenzen die Farbstoffe anregt, die anschließend mit einem Scanner detektiert werden. In Abbildung 2 sind die einzelnen Schritte zum Durchführen eines "Expression-Profiling"-Experiments mit einem cDNA-Array schematisch dargestellt [SELZER (2004), S. 141-143; MÜLHARDT (2009) S. 125f].

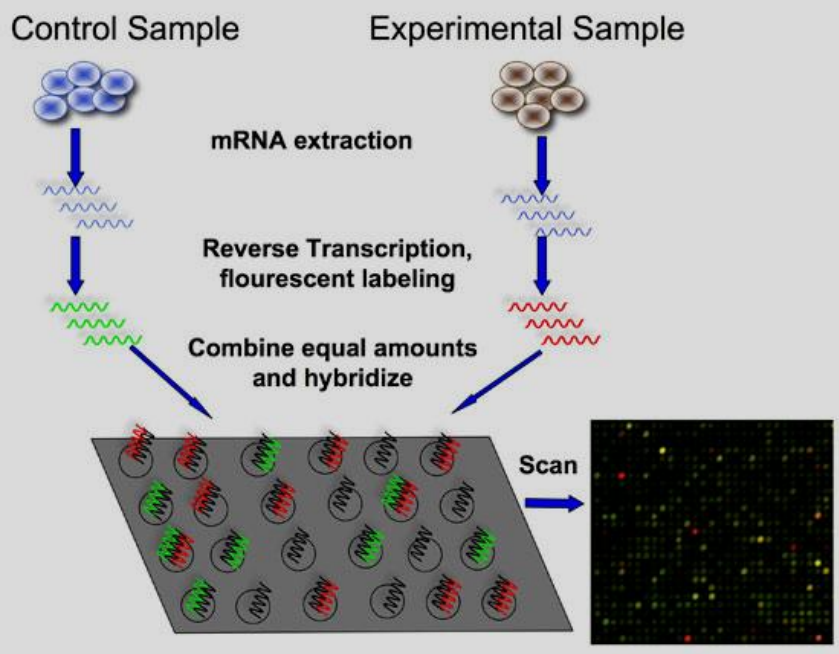


Abbildung 3 - "Expression-Profiling"-Experiment mit einem cDNA-Array [URL-7]
Aus Kontrolle und Probe wird die mRNA extrahiert und dann mittels Reverse-Transkriptase zu cDNA transkribiert. Danach wird die cDNA mit grün bzw. rot fluoreszierenden Farbstoffen markiert und auf den cDNA-Microarray aufgebracht. Nach der Hybridisierung der cDNA mit Oligos wird die restliche cDNA weg gewaschen. Die Anregung zur Emission der Farbstoffe erfolgt mittels Laser und anschließend werden die Signale gescannt.

1.6 Auswertung eines "Expression-Profiling"-Experiments

Das Prinzip der Microarrays ist recht einfach, aber die Analyse der Ergebnisse ist dafür umso komplexer. Grund hierfür sind die vielen Fehlerquellen, die während eines Microarray-Experimentes auftreten können. Man unterscheidet hier in statistische und systematische Fehler. Auf statistische Fehler hat man keinen Einfluss, da sie auf zufälligen Schwankungen beruhen. Demgegenüber stehen systematische Fehler, die durch eine falsche Kalibrierung von Messgeräten entstehen können. Zu den Messgeräten zählen z. B. der Laser oder der Scanner für die Fluoreszenzmessung. Aber auch durch sich ändernde Umweltbedingungen bzw. Laborbedingungen (z. B. Temperaturschwankungen oder Änderung der Luftfeuchtigkeit) während des Versuchsablaufs, können systematische Fehler auftreten. Statistische Fehler lassen sich durch mehrfaches Wiederholen des Experimentes minimieren. Dagegen können

systematische Fehler durch einen durchdachten Versuchsaufbau oder durch Kontrollexperimente reduziert werden. Ein Beispiel für ein solches Kontrollexperiment ist das sogenannte "Dye Swapping". Dabei werden die Cy3 und Cy5-Markierungen der gleichen cDNA-Ansätze vertauscht. Somit lässt sich herausfinden, ob bei der Markierung ein systematischer Fehler auftritt, der dann bei der Analyse berücksichtigt werden kann [SELZER (2004), S. 144].

Die Analyse der vom Microarray-Scanner erstellten Bilder ist der schwierigste Schritt. Denn dafür müssen die Intensitäten jedes einzelnen Spot, der vielen tausend Spots, eindeutig identifiziert und in nummerische Werte umgewandelt werden. Dazu müssen die Randbereiche der Spots eindeutig zuordenbar sein, sowie die Fluoreszenzintensitäten in beiden Kanälen (rot und grün) messbar und mit dem Hintergrund vergleichbar sein. Aufgrund der großen Anzahl von Microarray-Anbietern, mit teils unterschiedlichen Versuchsdurchführungen ist es nur logisch, dass Microarray-Daten systematische Fehler beinhalten. Zu den systematischen Fehlern zählen u.a. die ungleichmäßige Verteilung der Hybridisierungslösung auf dem Array oder verschiedene Halbwertszeiten der Fluoreszenzfarbstoffe. Daher ist eine Normalisierung der Daten unabdingbar, welche die Vergleichbarkeit der Experimentergebnisse gewährleistet. Denn die Experimente können an verschiedenen Tagen, bzw. in verschiedenen Laboren durchgeführt worden sein. Für die Normalisierung gibt es zahlreiche Algorithmen, die alle Vor- aber auch Nachteile besitzen [SELZER (2004), S. 145].

Der nächste Schritt der Datenauswertung ist es herauszufinden, welche Gene sich in beiden Proben, in ihrer Expression signifikant unterscheiden. Außerdem ist es im Interesse der Wissenschaftler bestimmte Genexpressionsmuster zu erkennen. Die Grundlage dafür ist, dass Gene die einem bestimmten Pathway angehören, oder gemeinsam auf bestimmte Umwelteinflüsse reagieren, koreguliert sind. Diese sind durch ein ähnliches Genexpressionsprofil gekennzeichnet. Die Methode um diese Muster einzuteilen, heißt Clustering und wird von Genen mit ähnlichen Expressionsprofilen durchgeführt [SELZER (2004), S. 146].

1.7 Genexpressionsdatenbanken und der MIAME-Standard

Jedes "Expression-Profiling"-Experiment generiert eine riesige Menge an Datensätzen. Ein Experiment kann mittels dutzender Microarrays durchgeführt worden sein, die jeweils tausende Spots besitzen. Dadurch werden sehr schnell Millionen von Messwerten erzeugt. Diese gilt es, in geeigneter Weise, zu verwalten und zu

analysieren. Daher mussten bioinformatische, wohl strukturierten Datenbanken aufgesetzt werden. Diese Datenbanken dienen der Speicherung und logisch Verknüpfung der Daten, um sie je nach biologischer Fragestellung abrufbar und auswertbar zu machen. Zu den Datenbanken, welche Genexpressionsdaten beinhalten zählen u.a. die "Gen Expression Omnibus"-Datenbank des NCBI (GEO) und die ArrayExpress-Datenbank des EBI, die beide als Quelle der Genexpressionsdaten für die vorliegenden Arbeit benutzt wurden [SELZER (2004), S. 148].

Die GEO-Datenbank ist unter der Webadresse http://www.ncbi.nlm.nih.gov/geo/ verfügbar. GEO dient als eine öffentlich zugängliche Sammlung für eine Vielzahl von Datensätzen aus Hochdurchsatz-Experimenten, bei denen u.a. das Transkriptom anhand der Häufigkeit der mRNA in unterschiedlichen Zellen gemessen wurde. Weiterhin sind auf der GEO-Datenbank Daten von anderen Hochdurchsatzverfahren zu finden, die nicht auf Microarrays basieren, sowie Daten die durch die serielle Analyse der Genexpression (SAGE) und durch Proteinidentifikations-Technologien erzeugt wurden. SAGE ist eine experimentelle Technik, die zur Analyse der Genexpression von Zellen oder Geweben eingesetzt wird. Weiterhin sind mehrere Tools zur Visualisierung und der weiteren Untersuchung der Daten in GEO integriert, wie z. B. hierarchische geclusterte "Heat Maps". Adäquate GEO-Datensätze sind zusammengefügt worden zu biologisch und statistisch vergleichbaren Dateien. Diese Dateien wurden dann indexgebunden in Entrez geladen, der Metasuchmaschine des NCBI. Dort sind sie zugänglich unter dem Punkt 'Gene' als 'GEO Profiles' und 'GEO DataSets'. Die GEO DataSets-Website ist unter der Web-Adresse http://www.ncbi.nlm.nih.gov/gds/ erreichbar [URL-8].

ArrayExpress ist ebenfalls öffentlich aber eine reine Sammlung von Microarray basierenden Genexpressionsdaten, welche dort im MIAME-Standard vorliegen. Das ist ein Annotationsstandard für Daten aus Microarray Experimenten. Dieser Standard beinhaltet eine eigene Markup Language - die "Microarray Gene Expression Markup Language" (MAGE-ML) - und wurde gemeinsam von der "Microarray Gene Expression Data" (MGED) Gesellschaft und der "Object Management Group" (OMG) entworfen. ArrayExpress hat drei hauptsächliche Ziele:

- Der weltweiten Wissenschaftsgemeinschaft ein Datenbank zur Verfügung zu stellen, auf die wissenschaftliche Publikationen verweisen können.
- Um der Gemeinschaft einen einfachen Zugriff auf hochwertige Expressionsdaten in einem Standardformat zu gewährleisten.

- Und um den Austausch von Bedingungen und Protokollen aus Microarray-Experimenten zu erleichtern.

Für ArrayExpress werden drei Arten von Einreichungen akzeptiert: Arrays, Experimente und Protokolle (einschließlich experimenteller Protokolle und Protokollen der Datenverarbeitung). Jedes davon kann einzeln eingereicht werden und bekommt einen eindeutigen Zugriffsschlüssel. Dieser kann später als interner oder externer Verweis dienen. Um die Daten an ArrayExpress zu übermitteln gibt es zwei Wege. Entweder direkt als MAGE-ML-Datei oder über das webbasierte Einsendungsformular MIAMExpress. Das Generieren von MAGE-ML-Datei erfordert ein "Laboratory Information Management System" (LIMS) und informatische Unterstützung. Zurzeit wird unter Mitwirkung des "Wellcome Trust Sanger Instituts" daran gearbeitet eine MIAME-konforme Pipeline zu etablieren. Ähnliche Pipelines, bspw. von "The Institute for Genomic Research" (TIGR) und Affymetrix, sind in der Testphase oder stecken noch in der Entwicklung. MIAMExpress ist aktuell in der Version 1.0 verfügbar. Es ist ein generisches Annotations-Tool, welches für die Annotation jeglicher Genexpressionsexperimente, die mittels Microarrays durchgeführt werden, geeignet ist. Da immer mehr Labore den MAGE-ML und MIAME-Standard anerkennen, nimmt die Menge der eingereichten Daten in ArrayExpress rasch zu. Dort sind dann diverse Suchanfragen nach Experiment, Spezies, Autor, Organisation, Array oder Zugriffsnummer möglich. Relevante Ergebnisse können folglich an das webbasierte Tool des EBI, den "Expression-Profiler", exportiert werden [URL-9].

1.8 Drug-Target-Screening und individuelle Medizin

Die Entwicklung neuer Medikamente hat in den letzten Jahren drastisch zugenommen. Deshalb liegen etliche Millionen von potenziellen Medikamenten in Lagern, deren Produzenten darauf warten, dass sie diese zulassen können. Um somit die hohen Entwicklungs- und Produktionskosten der jeweiligen Medikamente decken und einen Gewinn erzielen zu können. Darum ist das Auffinden von Medikamenten, die mit hoher Effizienz wirken und möglichst wenige, oder sogar keine Nebenwirkungen besitzen, von hohem Interesse für die Medizin. Natürlich liegt es auch im Interesse der Pharmaindustrie die Produktionskosten zu senken und die Entwicklung neuer Medikamente effizienter zu gestalten, um diesen Kostenvorteil letztendlich auch an das Krankenkassensystem sowie die Patienten weiter geben zu können [MÜLLER (2004), S. 164f].

Als potenzielle Zielmoleküle eines neuen Wirkstoffs werden Proteine hochgehandelt. Man geht davon aus, dass mindestens 10 % der Proteine, die aus dem menschlichen Genom exprimiert werden, mögliche Bindungspartner für die Wirkstoffe sein könnten. Als Wirkstoffe werden im Allgemeinen kleine Moleküle beschrieben, die als "Smal Molecular Drugs" (SMDs) bezeichnet werden. Die Zielmoleküle für diese SMDs werden im Allgemeinen Sprachgebrauch als "Drug Targets" betitelt. Der Begriff "Drug-Target-Screening" bezeichnet daher das Suchen nach geeigneten Drug-Targets für SMDs. Um das Ziel einer individualisierten Medizin zu erreichen ist es notwendig die entsprechenden Drug Targets (z. B. Rezeptoren) im Wirkungsort der Patienten (z. B. einem speziellen Gewebe) zu untersuchen, um beim Nachweis dieser die richtigen Wirkstoffe einsetzen zu können [MÜLLER (2004), S. 164f].

Zukunftsstudien weisen die individualisierte Medizin, welche häufig auch den Beinamen der personalisierte Medizin trägt, als eine Entwicklung aus, die durch das Zusammenwirken medizinischer und technologischer Bereiche angetrieben wird. Zu diesen Bereichen zählen die Deckung des medizinischen Bedarfs, die technologische Entwicklungen der Genom- und Postgenomforschung, sowie die Stärkung der Stellung von Patientinnen und Patienten im Gesundheitssystem. Die individualisierte Medizin zielt darauf ab, klinisch relevante personenbezogene Unterschiede zu identifizieren, um sie anschließend differenziert behandeln zu können. Als Indikatoren für diese Unterschiede fungieren Biomarker. Das sind Messgrößen zur Charakterisierung von normalen und krankhaft veränderten biologischen Prozessen. Vor allem solchen Biomarkern, die erstmals durch Hochdurchsatz-Techniken, wie die DNA-Microarrays, entdeckt oder bestätigt werden, wird eine besondere Bedeutung zugesprochen. Besonders wenn diese eine spätere Einteilung von Patientenpopulationen in medizinisch relevante Untergruppen (Strata) ermöglichen. Strata sind Patientengruppen, in die Patienten nach bestimmten Kriterien eingeteilt werden, die einer unterschiedlichen Behandlung bedürfen. Ziel ist es dadurch eine höhere Qualität der Versorgung zu erreichen, um damit das Risiko von Nebenwirkungen und Fehlbehandlungen zu reduzieren und um letzten Endes auch Kosten einsparen zu können [HÜSING (2012), S. 728].

2 Material

2.1 Die IDE Lazarus

Lazarus ist eine integrierte Entwicklungsumgebung (IDE) für die Programmiersprache Object Pascal. Die Entwicklung von Lazarus begann 1999 als eigenständiges Projekt. Lazarus arbeitet mit dem Free Pascal Compiler (FPC), der als unabhängiges Projekt aus Free Pascal entstand, dessen Entwicklung 1993 begann. Mittlerweile existiert er in einer 32 Bit- und einer 64 Bit-Version für Windows 32, Windows 64, Linux, FreeBSD, MacOS, DOS, OS/2 und für weitere Betriebssysteme. Unterstützt werden die Prozessorarchitekturen: Intel x86, AMD64/x86_64, PowerPC, PowerPC64, Sparc und Arm. Der FPC wird als Back-End bezeichnet. Back-End bedeutet, dass die Schnittstelle des Compilers näher am System ist, und im Gegensatz dazu ist Front-End näher an der Schnittstelle zum Benutzer. Die Kombination aus Back-End und IDE hat den Vorteil, dass der Quelltext effizient und hochoptimiert nativ, heißt vererbbar in weitere Klassen ist. Dies ist ein Vorteil im Gegensatz zu Java- und C#-IDEs, die Bytecode produzieren, der dann noch von einer Laufzeitumgebung interpretiert oder von einem Compiler in einen wenig optimierten Code, umgewandelt werden muss [CANNEYT (2011), S. 15f].

Ein großer Vorteil der IDE Lazarus ist ihre RAD-Fähigkeit (Rapid Application Development), also die Möglichkeit grafische Programme einfach und schnell zu programmieren. Hinterlegt ist dieser IDE der FPC, der zum Großteil kompatibel zu Delphi ist. Der wohl größte Vorteil von Lazarus und dem FPC ist seine Plattformunabhängigkeit. So läuft Lazarus auch unter Linux, MacOS, FreeBSD und noch weiteren Plattformen. Die Bibliothek für grafische Komponenten heißt in Lazarus "Lazarus Component Library" (LCL). Viele Units, Klassen und Bezeichner sind genauso wie bei Delphi deklariert, womit eine Portierung von Delphi-Code nach Lazarus möglich ist. Einen Nachteil den Lazarus besitzt, ist die nicht ganz vollständige Kompatibilität zur "Visual Component Library" (VCL), die u.a. in Borland Delphi und C# verwendet wird, und zum vereinfachten Entwurf von Windows-Anwendungen entwickelt wurde [CANNEYT (2011), S. 15; URL-10].

Im Gegensatz zu Delphi ist Lazarus eine komplette Open Source-IDE, auch für kommerzielle Anwendungen als Freeware verfügbar, und lässt sich auf mehr als 15 Plattformen installieren. Dank der Open-Source-Gemeinschaft wurden schon viele

Delphi-Bibliotheken auf Lazarus portiert, sowie neue Bibliotheken exklusiv für Lazarus geschrieben. Eine umfangreiche Hilfe zur Programmierung mit Object Pascal, findet sich auf http://www.delphi-treff.de, sowie in der Online-Hilfe von Lazarus. Mit Lazarus lassen sich native grafische Anwendungen, ebenso wie Windows-Dienste, Konsolenwerkzeuge, Bibliotheken, Datenbankanwendungen, Webanwendungen und vieles Weitere programmieren. Des Weiteren unterstützt Lazarus die sogenannte Crosskompilierung, so können bspw. Windows-Programme unter Linux erzeugt werden [CANNEYT (2011), S. 16f].

Bei Lazarus gibt es ein Package-Konzept, mittels dessen mehrere Units zu einer größeren Einheit zusammengefasst werden können. Ein Package ist eine Dateisammlung mit allen Informationen, die zum Erstellen eines Programms benötigt werden. Diese Informationen beinhalten, wie man das Package kompiliert (Compilereinstellungen) und welche anderen Packages dafür benötigt werden. Somit ist es möglich, dass Source Code schnell und unkompliziert auf andere Computer, mit anderen Datei-Pfaden und Betriebssystemen übertragen werden kann [CANNEYT (2011), S. 17].

2.2 Objekt Pascal

Object Pascal ist eine Sammelbezeichnung für mehrere Dialekte der Sprache Pascal, die um die Objektorientierte Programmierung erweitert wurden. Die bekannteste dieser Sprachdialekte ist der Pascal-Dialekt der Entwicklungsumgebung Delphi, die vom Unternehmen Borland entwickelt wurde. Object Pascal ist als höhere Programmiersprache anzusehen, da sie viel mehr an die natürliche (englische) Sprache als an die Maschinensprache angelehnt ist. Eine der Grundprinzipien, auf denen die Arbeitsweise von Computern beruht, ist die sequentielle und endliche Ausführung von Befehlen durch den Prozessor. Der Object Pascal Code ergibt für den Rechner vorerst keinen Sinn, bis er vom FPC übersetzt wird. Die Übersetzung des Codes mittels dem FPC erfolgt im Normalfall sehr schnell. Delphi und Lazarus bieten eine Menge von vorgefertigten Komponenten an. Weitere können unter Delphi hinzugekauft oder selbst programmiert werden. Für Lazarus hält die Open-Source-Gemeinschaft einige Komponenten zum freien Download bereit, die aus Delphi portiert wurden. Diese Komponenten sind ebenfalls mit Object Pascal geschrieben [URL-12; BINZINGER (2006) S. 20, 23f].

2.3 Vorhandene Datenstruktur

Die zentrale Datenstruktur des XpressionXplorers basiert auf den *.CEL-Dateien aus den Datenbanken GEO und ArrayExpress. Aus den Datenbanken wurden aber nur *.CEL-Dateien benutzt, die von Microarray-Experimenten abstammen, die mit dem "GeneChip® HG U133 Plus 2.0" von Affymetrix durchgeführt wurden. Jedem Microarray-Experiment, bei dem eine bestimmte Zelle auf ihre Genexpression hin untersucht wurde, wird ein eindeutiger Zugriffsschlüssel zugeordnet. Dies ist sozusagen der primäre Schlüssel für alle Datenbankeinträge in der GEO- und in der ArrayExpress-Datenbank. Ein Beispiel für so einen Zugriffsschlüssel ist: GSM449176. Der Eintrag davon in der GEO-Datenbank ist in Abbildung 4 dargestellt.

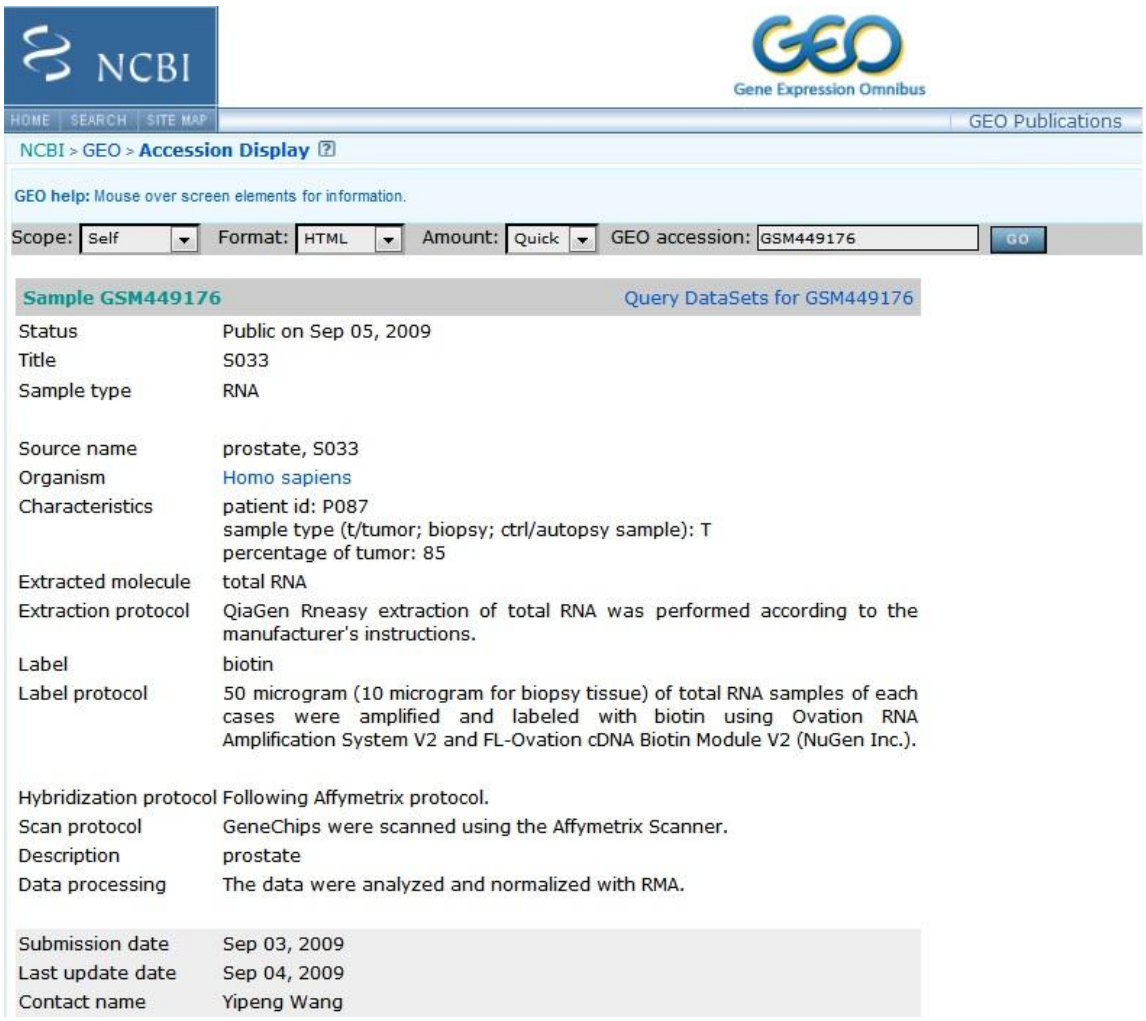


Abbildung 4 - Datensatz eines Microarray-Experiments in der GEO-Datenbank [URL-11]
Die Abfrage der GEO-Datenbank erfolgte mit dem Zugriffsschlüssel GSM449176. Angezeigt werden u.a. Informationen zur Publikation, zur Quelle, aus der die Probe stammt, und zur Durchführung des Microarray-Experiments (Informationen aus den Protokollen zur Herstellung des DNA-Ansatzes und der Hybridisierung auf dem Microarray).

Die Daten des Microarray-Experimentes lassen sich als *.CEL-Datei herunterladen. Diese *.CEL-Dateien wurden für den XpressionXplorer unter Verwendung der freien Software RMAExpress zusammengefasst, normalisiert und korrigiert. Dadurch entstanden Dateien mit der Dateiendung *.rma. Diese wurden schließlich nach Gewebeart, aus der die Probe stammt, klassifiziert und danach zu Dateien mit der Endung *_mp.txt zusammengefasst. Diese *_mp.txt-Dateien sind im Gegensatz zu den *.CEL-Dateien mittels Texteditoren, sowie mittels Software für Tabellenkalkulationen, lesbar. Ein Screenshot des Programms RMAExpress lässt sich in folgender Abbildung 5 betrachten.

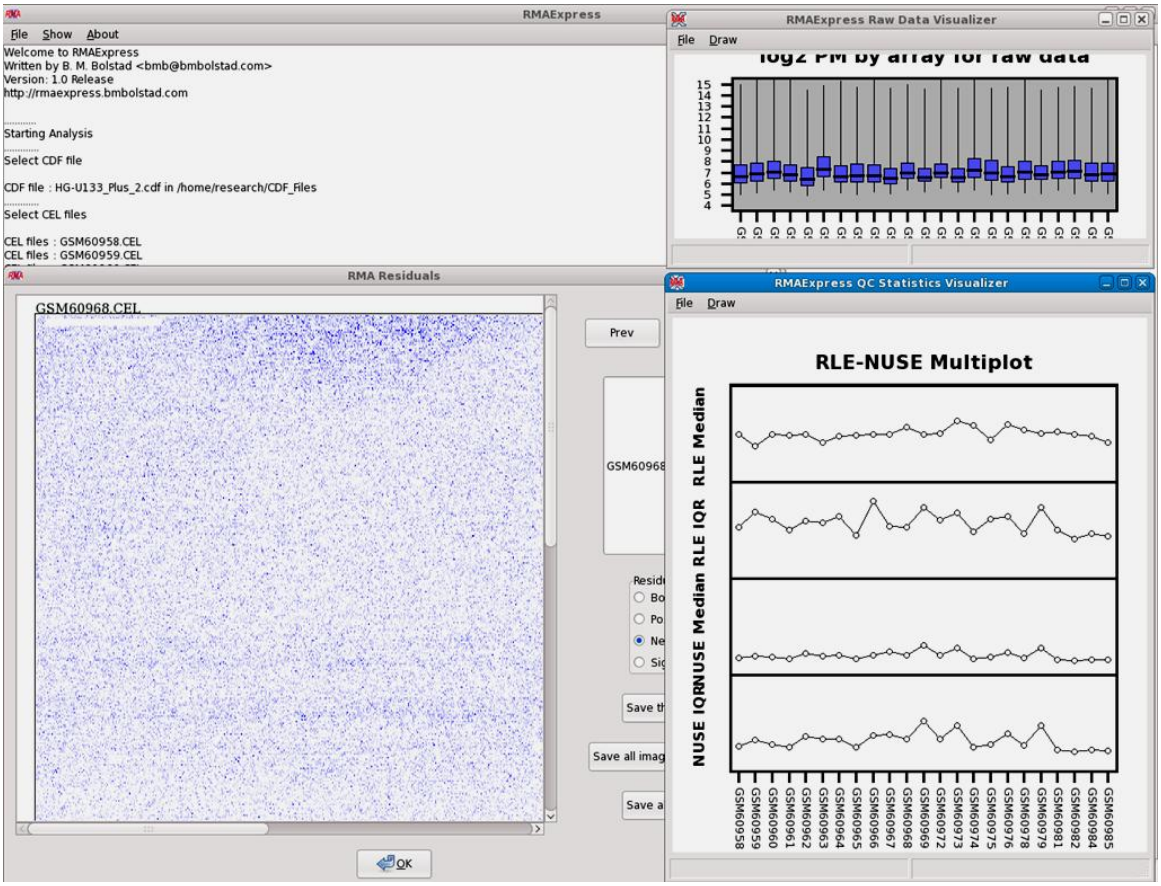


Abbildung 5 - Screenshot des Programms RMAExpress [URL-14]
Das freie Programm RMAExpress zum Verarbeiten von *.CEL-Dateien aus Genexpressions-datenbanken. Das Fenster oben links ist das Hauptfenster, in dem die Dateien eingelesen werden. Danach können die Genexpressionsdaten in den weiteren Programmkomponenten bearbeitet werden. Informationen, die für eine weitere Analyse relevant sind, lassen sich somit herausfiltern.

Jede der zusammengefassten, normalisierten und korrigierten *_mp.txt-Dateien besteht aus einer Tabelle mit 54.675 Zeilen und mehreren tausend Spalten. Jede Zeile beschreibt ein Probeset, das in Korrelation zu einem humanen Gen steht. Jede Spalte, ab der sechsten, beschreibt ein Microarray-Experiment. Somit stellen die Spalten die

Genexpressionsprofile aus einer logisch verknüpften Gruppe von Microarray-Experimenten dar. Die logische Verknüpfung beruht darauf, aus welchem Gewebe, bzw. aus welcher Körperflüssigkeit (z. B. Blut) die Proben für die Experimente entnommen wurden. Diese *_mp.txt-Dateien werden durch die erste Programmkomponente eingelesen, die beim Starten des XpressionXplorers ausgeführt wird.

Außerdem existiert eine Datei, die die Baumstruktur der 'Treeview' abbildet. Diese muss zuerst eingelesen werden, um die Baumstruktur der VirtualTreeview-Komponente zu setzen. Diese Datei heißt 'treeData.txt' und beinhaltet die Bezeichnung der Zweige des Baumes ('branches') und die Ebene der Zweige. Die Ebenen der Zweige werden durch die vorangestellten Tabulatoren bzw. Leerspalten festgelegt. In folgender Abbildung 6 ist ein Ausschnitt aus der Datei zu sehen.

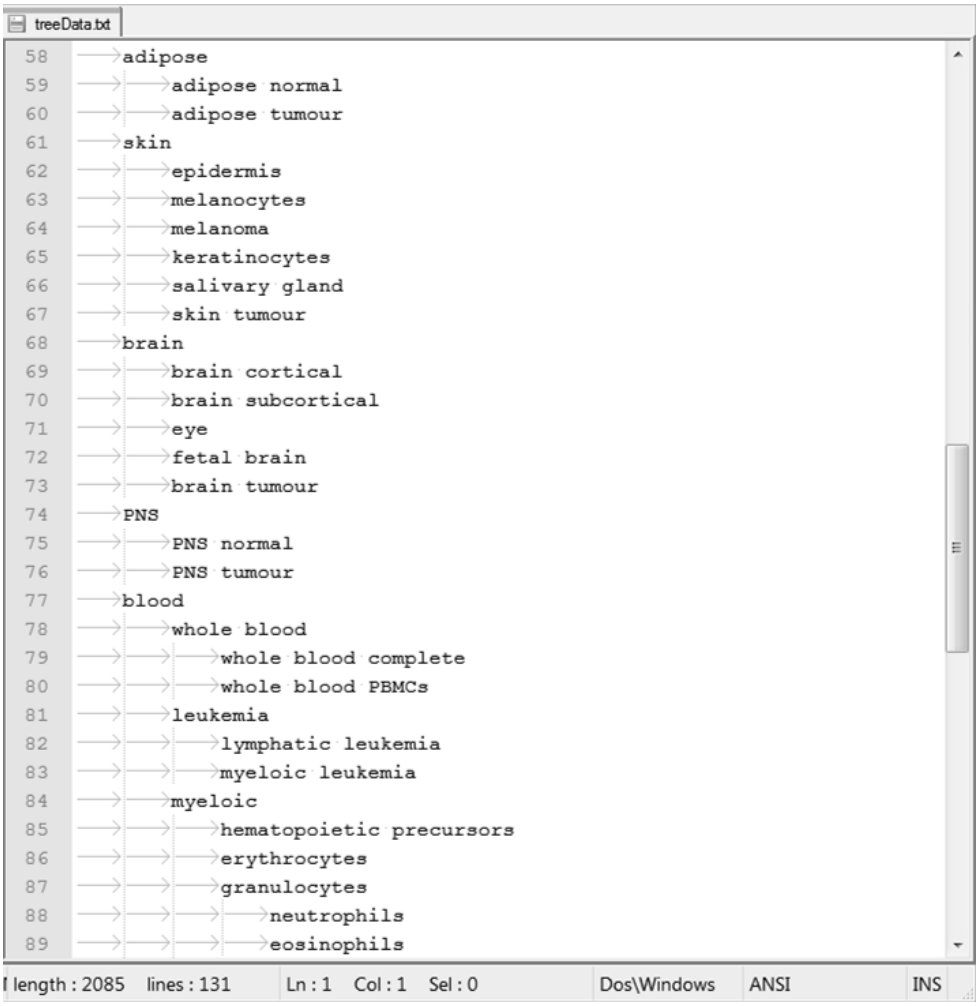


Abbildung 6 - Ausschnitt aus der Baumstrukturvorlage
Zeile 58 bis 89 aus der Datei 'treeData.txt', die als Vorlage für die Baumstruktur der VirtualTreeview dient. Man sieht die Einträge für Fett-, Haut-, Gehirn-, Blutzellen, sowie die Zellen des peripheren Nervensystems (PNS) und deren Unterverzweigungen.

Desweiteren sind drei Dateien vorhanden, welche Informationen über die GeneChips® enthalten. Aus diesen Dateien sollen nur spezielle Attribute ausgelesen werden und diese in der virtuellen Baumansicht dargestellt werden. Die drei Dateien mit den Chip-Attributen haben die Dateinamen: 'chipinfoTV.txt', 'chips_ausw.txt' und 'Decomposition Celltypes.txt'.

Da der MIAME-Standard zur Annotation zur Zeit leider oft nur theoretisch funktioniert, wurden einige Annotationen von tausenden Microarray-Experimenten manuell in eine Excel-Tabelle eingetragen und hierarchisch nach der Annotation 'description' sortiert. Die dabei entstandene Datei wurde als 'chipinfoTV.txt' bezeichnet. In Abbildung 7 ist ein Ausschnitt aus der 'chipinfoTV'-Datei mit den Daten zu sehen, die für die VirtualTreeview von Bedeutung sind. Die benötigten Attribute für die "Virtual String Tree"-Komponente (VST) sind die dritte und fünfte Spalte mit den Einträgen für Farbe und 'branch'. Farbe bezeichnet eine Systemfarbe der VCL, wie bspw. clwhite oder clblack. Diese Farbe ist für den XpressionXplorer von Bedeutung, um die Farbdarstellung der Chip-Experimente in einem Diagramm (z. B. Scatterplot) darstellen zu können. Im Allgemeinen werden die Microarray-Experimente als Chips bezeichnet, was sich aus der Kurzbezeichnung GeneChip® der DNA-Microarrays von Affymetrix ableitet. Der 'branch' beschreibt die Position des GeneChips® in der Baumstruktur. Also entspricht jeder String 'branch' aus der 'chipinfoTV.txt' immer einem zugehörigem String 'branch' in der 'treeData.txt' und ermöglicht somit eine eindeutige Zuordnung zu den Zweigen des Baumes.

id-	description	farbe	ausgabe	branch	b
GSM707916	adipocytes disease Simpson-Golabi-Behmel sync	clwhite	rest3	adipose normal	
GSM601629	adipocytes from msc no TNF no insulin 9	clwhite	rest3	adipose normal	
GSM239379	adipocytes from MSCs	clwhite	rest3	adipose normal	
GSM239457	adipocytes from MSCs	clwhite	rest3	adipose normal	
E-MEXP-858-CW3	adipocytes from MSCs 7d after induction	clwhite	rest3	adipose normal	
E-MEXP-858-CW3	adipocytes from MSCs 9h after induction	clwhite	rest3	adipose normal	
GSM489176	adipocytes from MSCs fat brown	clwhite	rest3	adipose normal	
GSM601632	adipocytes from MSCs treatment: insulin	clwhite	rest3	adipose normal	
GSM601634	adipocytes from MSCs treatment: TNF	clwhite	rest3	adipose normal	
GSM601636	adipocytes from MSCs treatment: TNF + insulin	clwhite	rest3	adipose normal	
E-MEXP-1216-Pat	adipocytes from stem cells day 0	clwhite	rest3	adipose normal	
GSM458608	adipocytes from stromal/stem cells (ASC)	clwhite	rest3	adipose normal	
GSM458609	Adipocytes from stromal/stem cells (ASC) induc	clwhite	rest3	adipose normal	
GSM519604	adipocytes hASC pre-adipocyte, day 14	clwhite	rest3	adipose normal	
GSM707912	adipocytes disease Simpson-Golabi-Behmel sync	clwhite	rest3	adipose normal	

Abbildung 7 - Ausschnitt aus der Datei 'chipinfoTV.txt'
Die Datei 'chipinfoTV.txt' - geöffnet mit Excel. In der ersten Spalte wird der Zugriffsschlüssel (in Programmstruktur auch als Chip-ID bezeichnet) angezeigt. In der zweiten Spalte ist die Beschreibung zu dem Microarray-Experiment, die aus der GEO-Datenbank stammt, dargestellt. Die wichtigen Attribute aus dieser Datei sind die in den Spalten 3 und 5 mit den Bezeichnungen 'farbe' und 'branch'.

In der Datei 'chips_ausw.txt' sind über 9.000 Datensätze von Microarray-Experimenten enthalten. Diese stellen eine Auswahl aus den Microarray-Experimenten dar, in der nur Chips eingetragen wurden bzw. werden, für die wirklich Daten vorliegen. D.h. die Datei enthält eine Zusammenfassung von Daten aus *.CEL-Files, die mit Hilfe von Programmen, wie RMAExpress oder ähnlichen erzeugt wurden. Die erste Spalte dieser Datei enthält ebenfalls wie bei der 'chipinfoTV.txt' die eindeutigen Zugriffsschlüssel jedes einzelnen Microarray-Experiments, die als Chip-ID deklariert sind. Die zweite Spalte enthält die 'chip description', deren Einträge mit denen der 'chip description' der GeneChips® in der 'chipinfoTV.txt' übereinstimmen. Weiterhin werden die Einträge der Attribute Eingabe- und Ausgabe-Datei zur Anzeige in der VST benötigt. Die Spalte Eingabe-Datei beinhaltet die *.rma-Dateinamen, die nach der Verarbeitung der *.CEL-Dateien durch RMAExpress entstanden sind. Die Spalte Ausgabe-Datei beinhaltet die *_mp.txt-Dateinamen, zu denen die jeweiligen *.rma-Dateien zusammengefasst wurden. Ein weiteres Attribut in der Datei 'chips_ausw.txt' ist die "Summe Degradation ACTB" die reelle Zahlenwerte beinhaltet, die die Qualität der Microarray-Experimente beschreiben. Denn die Qualität jedes Microarray-Experiments ist abhängig von Faktoren, wie der Qualität und der Quantität der Gesamt-RNA. Denn die RNA kann während des Experiments bereits Schaden nehmen und damit folglich teilweise zerstört (degradiert) sein, oder nur in sehr geringen Mengen vorliegen. In nachfolgender Tabelle 3 ist die Bedeutung der jeweiligen Werte des Attributs "Summe Degradation ACTB" dargestellt.

Tabelle 3 - Wert zur Qualität von Microarray-Experimenten

Wert des Attributs "Summe Degradation ACTB"	Aussage über die Qualität des Microarray-Experiments
< 5	hervorragend
5 - 10	gut
10 - 15	beeinträchtigt
> 15	problematisch

Die GeneChip®-Datensätze mit den problematischen Werten werden nicht herausgefiltert, da sie oft aus sehr interessanten Microarray-Experimenten mit seltenen Zelltypen stammen, die im Vorfeld sehr aufwendig aufgereinigt werden mussten.

2.4 Vorhandenes Programm: XpressionXplorer

2.4.1 Aufbau des XpressionXplorers

Beim Starten des XpressionXplorers wird als erstes ein einzelnes Fenster geöffnet. In diesem Fenster gibt es ein Menü mit den Einträgen 'File', 'Run' und 'Show' und eine Nachrichten-Box, die den aktuellen Status anzeigt. Auf folgender Abbildung ist das Startfenster des XpressionXplorers zu sehen.

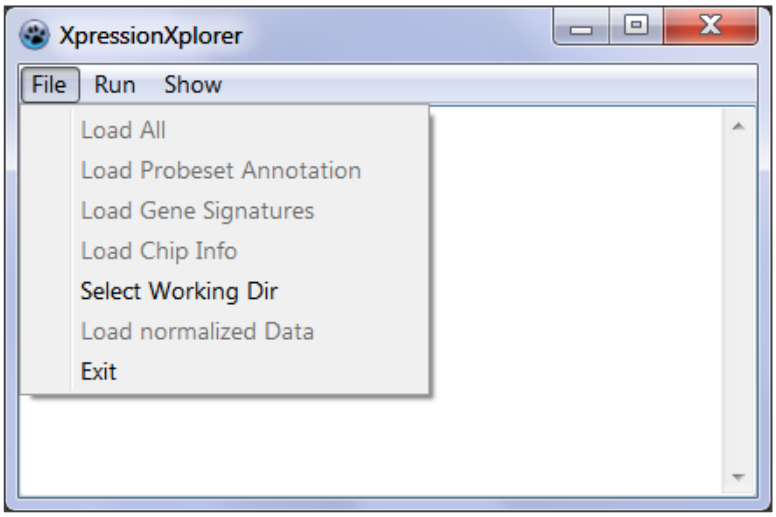


Abbildung 8 - Startfenster des XpressionXplorers
Das schlichte Startfenster des XpressionXplorers mit einem Menü und einer Nachrichtenbox, zum Anzeigen des aktuellen Status des Ladevorgangs.

Im Menüeintrag 'File' ist zuerst das Arbeitsverzeichnis zu wählen, in dem sich die Dateien mit den Daten zum Einlesen der Datenstruktur befinden müssen. Die Datenstruktur umfasst, zu denen in Abschnitt 2.3 beschriebenen Dateien, noch weitere. Diese sind allerdings für die zu verbessernde Programmkomponente unwichtig. Die wichtigsten Dateien sind die normalisierten *._mp-Dateien, welche die Expressionswerte aller 54.675 Probes zu den zu analysierenden Microarray-Experimenten beinhalten. Nach der Auswahl des Arbeitsverzeichnisses genügt ein Klicken auf den Menüeintrag 'Load All', um die Annotationen der Probesets, die Gensignaturen und schließlich auch die normalisierten *_mp.txt-Dateien zu laden. Das Laden einer Anzahl von 30 *_mp.txt-Dateien dauert etwa 1½ Minuten. Danach können unter dem Menüeintrag 'Show' die unterschiedlichen Analysetools gestartet werden (Abbildung 9). Die Programmkomponente 'Differential Expression' ist eines dieser Tools, um die unterschiedliche Genexpression in bspw. gesundem und krankem Gewebe zu vergleichen. Auf diese Art soll es möglich sein spezifische Biomarker zu finden, die das Genexpressionsprofil einer Krankheit charakterisieren. Ziel ist es u.a. damit neue Wirkungsorte für medikamentöse Therapieansätze zu erforschen.

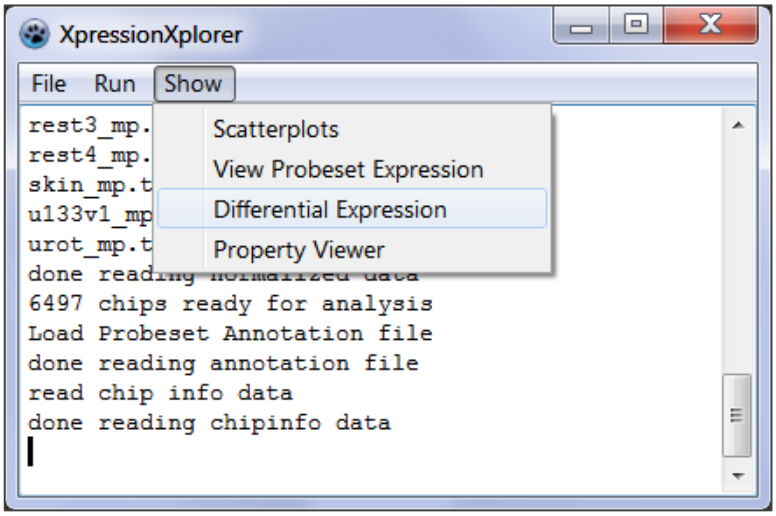


Abbildung 9 - Startfenster nach der Initialisierung
Nach Klicken auf den Eintrag 'Show' werden die Einträge angezeigt, die zum Starten der verschiedenen Komponenten ausgewählt werden können. Die Programmkomponente, welche weiterentwickelt wurde ist die mit der Bezeichnung 'Differential Expression'

2.4.2 Die alte 'Differential Expression'-Komponente

In dieser Programmkomponente werden die Datensätze der GeneChips® zu den Zweigen einer Baumstruktur zugeordnet. In der Baumansicht ('Treeview') lassen sich bestimmte Chips auswählen und jeweils zu einer von zwei Gruppen (A oder B) zuordnen. Nach dem Einstellen spezieller Parameter im "Graphical User Interface" (GUI)

der 'Differential Expression'-Komponente, lassen sich Gen-Listen berechnen. Diese dienen dem Anwender zum Vergleichen der Genexpressionsdaten aus verschiedenen Microarray-Experimenten, die zuvor zu den Gruppen A und B zugeordnet wurden. Diese Gen-Listen sind den Benutzern der Software beim Finden von Biomarkern behilflich. Ein Screenshot der alten Programmkomponente 'Differential Expression' ist auf der nächsten Seite in Abbildung 10 dargestellt.

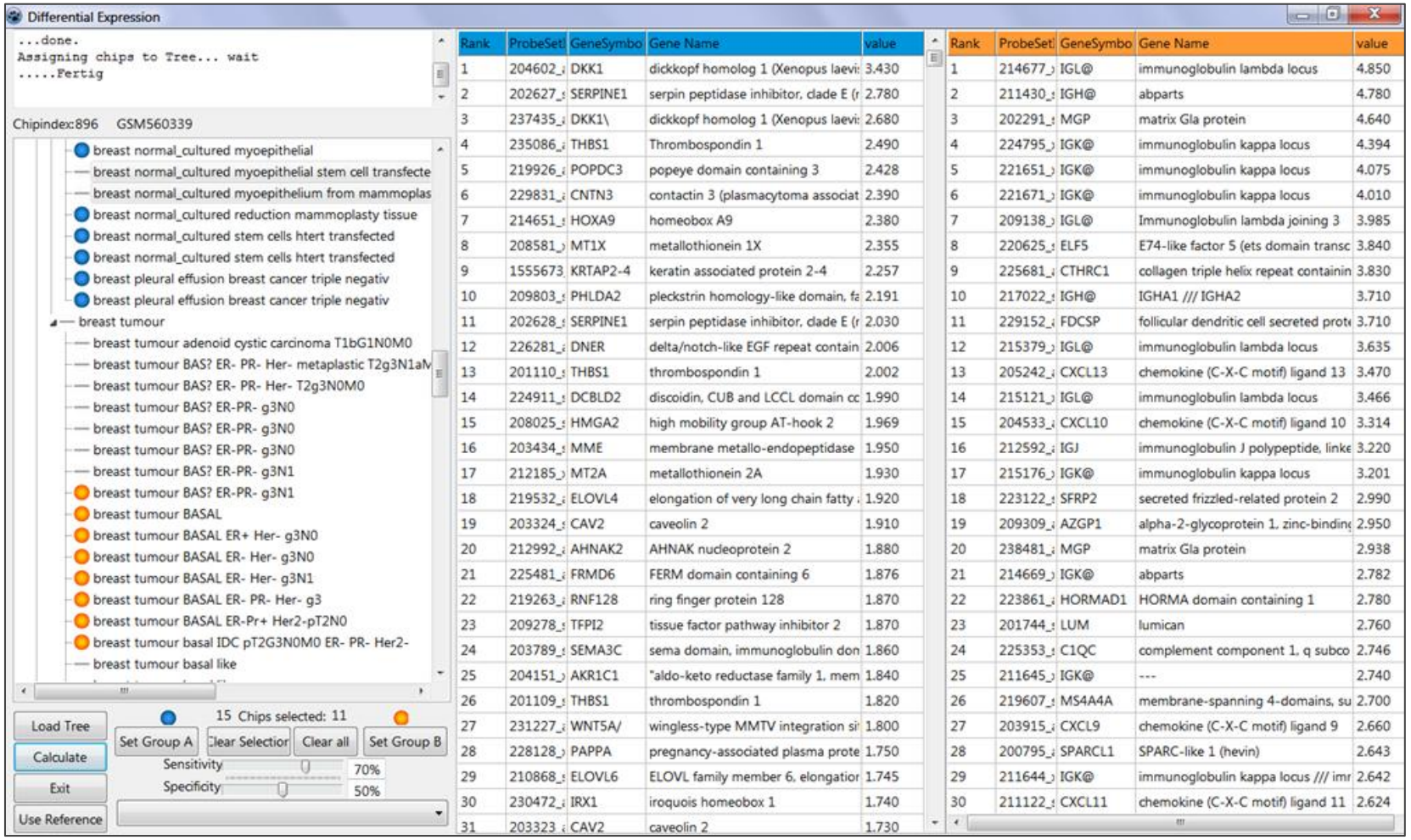


Abbildung 10 - Die Programmkomponente 'Differential Expression'
Im linken Bereich ist die Baumansicht mit den Beschreibungen der Chips zu sehen, von denen einige der Gruppe A oder B zugeordnet sind. Im rechten Bereich sind die Listen erkennbar, in denen die Probesets und Gene absteigend nach dem Wert 'value' geordnet sind**.**

2.4.3 Schwächen der alten 'Differential Expression'- Komponente

Die einfache Baumansicht zur Auswahl der Chips hat einige Schwächen, die ein effizientes Arbeiten erschweren. Dazu gehören folgende Mängel, die es galt mit der Entwickelung einer neuen Komponente zu beseitigen:

- Es ist keine Mehrfachauswahl der Chips möglich, weder durch Drücken der SHIFT-Taste noch durch ein Auswahlrechteck per Maus.
- Die Auswahl kann nur durch Klicken auf die Buttons 'Set Group A' bzw. 'Set Group B' erfolgen und nicht einfach durch ein Pop-up-Menü, das sich durch Drücken der rechten Maustaste öffnet.
- In der Baumansicht werden nur die Chip-Beschreibungen angezeigt, aber keine zusätzlichen biologischen und technischen Informationen zu den Microarray-Experimenten.
- Es existiert keine Suchfunktion, um bestimmte Einträge schneller zu finden.
- Es ist nicht möglich die Auswahllisten zu speichern und beim nächsten Programmstart zu laden. Somit müssen die Einträge umständlich jedes Mal von Neuem zusammengeklickt werden.
- Die Spalten der Listen lassen sich nicht vergrößern, wodurch die Bezeichnungen der Probesets, sowie die Namen der Gene nicht vollständig lesbar sind.

3 Methoden

Für das erfolgreiche Erstellen der neuen 'Differential Expression'-Programmkomponente war es vorerst notwendig den Aufbau des vorhandenen XpressionXplorer-Programms zu analysieren. Des weiteren mussten Recherchen zur Anwendungsentwicklung in Lazarus, sowie zum Arbeiten mit dem VirtualTreeview-Package gemacht werden. Dazu wurden Literaturrecherchen in Büchern und auf Webseiten zu Delphi und Lazarus betrieben. Ergänzend wurden interaktive Übungen durchgeführt, sowie Demos des VirtualTreeview-Packages ausprobiert. Um das VirtualTreeview-Package in Lazarus nutzen zu können, wurde dies von der Website http://www.lischke-online.de/index.php/all-downloads (verfügbar am 18.09.2013) von Mike Lischke heruntergeladen und in die IDE installiert. Dafür war es aber vorab erforderlich, ein weiteres Package mit der Bezeichnung 'lclextensions_package 0.5' zu installieren. Denn das VirtualTreeview-Package besitzt von diesem einige Abhängigkeiten.

3.1 Gestaltung eines neuen Formulars

Um die alte Programmkomponente 'Differential Expression' durch eine neue zu ersetzen, musste zuerst ein neues Formular gestaltet werden. Aus Platzgründen wurde für das neue Formular das Objekt 'TPageControl' verwendet. Dies ist eine Registerkartenkomponente aus der LCL-Klassenbibliothek [CANNEYT (2011), S. 491]. Auf dieser wurden zwei Registerkarten 'Virtual Treeview' und 'Gene Lists' angelegt. Die erste Registerkarte 'Virtual Treeview' wurde folgendermaßen gestaltet:

- Im oberen Bereich gibt es eine Nachrichten-Box (Memobox), die Informationen über den Status, während des Ladens des virtuellen Baumes anzeigt.
- In der Mitte wurde eine 'TVirtualStringTree'-Komponente (VST) angelegt, für die auf dem Formular der meiste Platz zur Verfügung gestellt wurde, da in ihr alle für die Nutzer relevanten Attribute der GeneChips® dargestellt werden.
- Am unteren Randbereich wurde ein Panel mit verschiedenen Buttons, zur Steuerung des VSTs angelegt.

In Abbildung 11 ist das Seitenverhältnis, der ersten Registerkarte zu sehen, welches gewählt wurde, um die drei Haupt-Komponenten auf dem GUI möglichst optimal

darzustellen. Dadurch soll es den Anwendern ermöglicht werden, das Fenster bei Bedarf zu maximieren, ohne dass sich die Komponenten willkürlich verschieben.

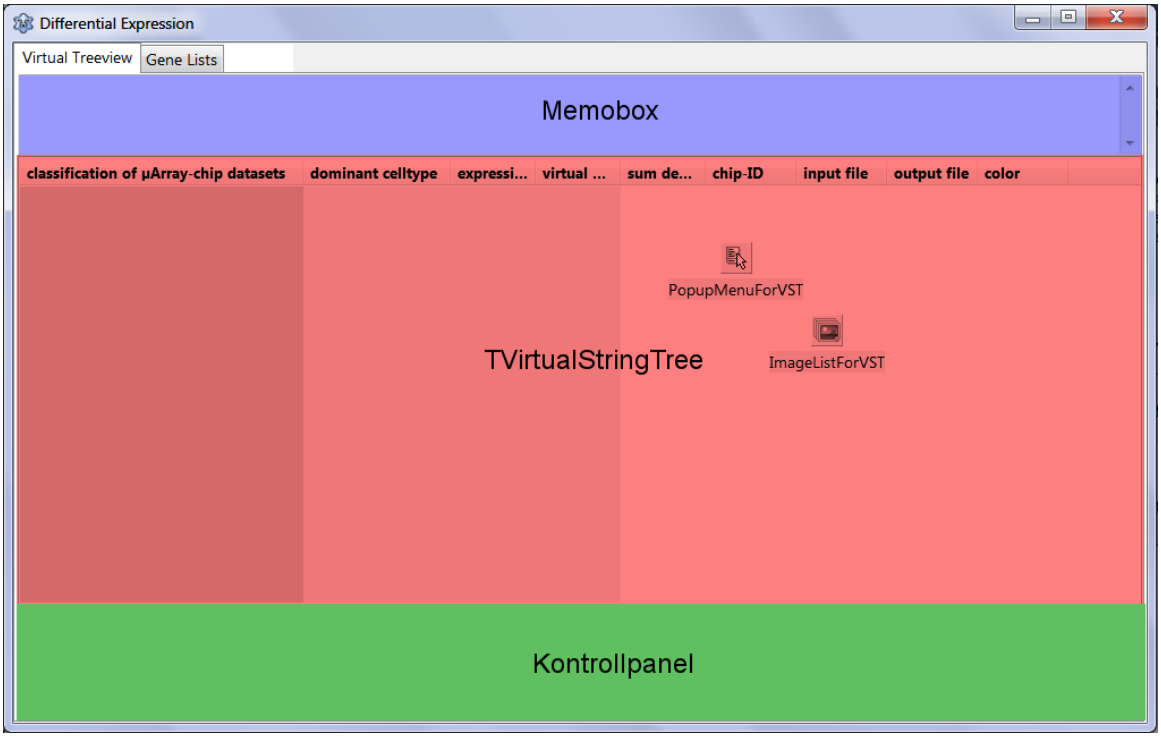


Abbildung 11 - Seitenverhältnis der Registerkartenkomponente 'Virtual Treeview'
Diese Abbildung zeigt das Seitenverhältnis der Registerkartenkomponente 'Virtual Treeview' des Formulars 'Differential Expression' aus dem XpressionXplorer. Das Nachrichtenfeld ist blau hinterlegt. Der Bereich für die virtuelle Baumansicht ist rot hinterlegt, zu der ein Pop-up-Menü und eine Imagelist gehören. Der Bereich zur Steuerung der VirtualTreeview unterhalb des VSTs wurde grün markiert.

Dem VST wurden neun Spalten hinzugefügt, um alle Chip-Attribute darstellen zu können, die für die Nutzer des XpressionXplorers nützlich sind. In der ganz linken Spalte wird die Baumstruktur erzeugt, zu deren Zweigen dann die Chip-Beschreibungen zugeordnet werden. In Spalte zwei bis vier (von links) werden die biologischen Daten und zu den restlichen Spalten die technischen Daten, nach dem Initialisieren der VirtualTreeview, angezeigt. Das Ereignis zum Initialisieren des virtuellen Baums wird durch Klicken auf den Button 'Load Tree' im Kontrollpanel ausgelöst, worauf im Abschnitt bereits 3.2 genauer eingegangen wurde. In der Imagelist sind zwei kleine Bilder im Format 16 x 16 Pixel hinterlegt. Auf diesen kleinen Bildern sind einmal ein grüner und einmal ein roter Punkt abgebildet. Diese werden für die Darstellung der Auswahl der Gruppen A und B durch den Nutzer verwendet. Die Gruppen A und B stellen zwei Auswahllisten von Chips dar, welche der Benutzer selbst in der virtuellen Baumansicht zusammenstellt. Dabei soll ihm das Pop-up-Menü behilflich sein, welches sich durch Drücken der rechten Maustaste auf den VST auslösen lässt. Das Pop-up-Menü ist

ebenfalls mit der Imagelist verknüpft, so dass der grüne Punkt vor dem Eintrag 'add to group A' und der rote Punkt vor dem Eintrag 'add to group B' dargestellt wird, wodurch eine bessere Übersichtlichkeit für den Nutzer geschaffen wird. Das Kontrollpanel liegt, in Anlehnung an die vorherige Programmkomponente, im unteren Bereich des Fensters. Es wurde erweitert um die Buttons 'Save Selection' und 'Load Selection' zum Speichern und Laden der Auswahllisten, außerdem um ein Editierfeld, sowie einen 'Search'-Button, zum Suchen nach speziellen Chip-IDs im VST.

Die zweite Registerkarte der Programmkomponente wurde mit der Bezeichnung 'Gene Lists' versehen. Sie dient dem Vergleich zweier, im VST ausgewählter Gruppen von Chips. In diesen soll nach Klicken auf den Button 'Calculate' im Kontrollpanel der Registerkarte 'Virtual Treeview' eine Gegenüberstellung der Gene erfolgen, welche den höchsten Expressionsfaktor besitzen. Die Eigenschaften der Listen wurden im Objektinspektor von Lazarus so angepasst, dass nun ein Vergrößern der Spaltenbreite durch den Nutzer erfolgen kann, um sich die Bezeichnung der Probeset und der Gene vollständig anzeigen zu lassen. In folgender Abbildung 12 ist ein Screenshot der zweiten Registerkartenkomponente mit der Bezeichnung 'Gene Lists' abgebildet.

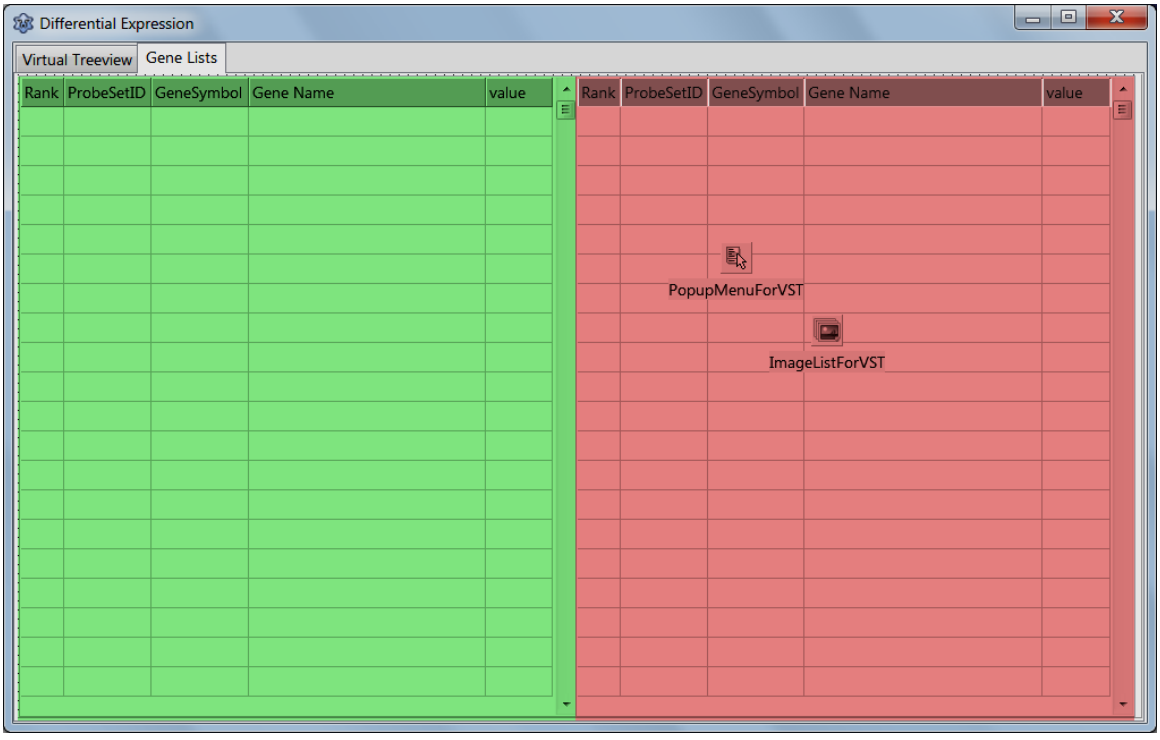


Abbildung 12 - Registerkartenkomponente 'Gene Lists'
Die beiden Komponenten vom Typ 'TStringGrid' sind grün und rot eingefärbt. In der grün eingefärbten Tabelle wird die Gen-Liste der Gruppe A, und in der rot eingefärbten Tabelle die Gen-Liste der Gruppe B geladen. Die Spalten stellen die Probeset-ID, das Gensymbol, den Gen-Name und den Expressionswert dar. Der größte Genexpressionswert steht nach der Berechnung auf dem ersten Rang.

Hinter dem Formular verbirgt sich die zugehörige Unit, welche den Quelltext beinhaltet. Denn Lazarus erzeugt automatisch für jedes Formular eine neue Unit. Die Unit des Formulars wurde als 'virtualtreeviewunit' bezeichnet.

3.2 OnClick-Ereignis zum Initialisieren der VirtualTreeview

Mit dem Schreiben der OnClick-Ereignisse in der 'virtualtreeviewunit' begann die hauptsächliche Programmierarbeit. Das OnClick-Ereignis des Buttons 'Load Tree' löst eine Reihe von Prozeduren und Funktionen aus. So musste etwa eine Prozedur zum Laden der Baumstruktur aus der Datei 'treeData.txt', sowie drei Prozeduren zum Initialisieren von Daten aus den Dateien 'chipinfoTV.txt', 'chips_ausw.txt' und 'Decomposition Celltypes.txt' geschrieben werden. In Abbildung 13 sind die Prozeduren zum Laden des virtuellen Baums, nach dem Klicken auf den Button 'Load Tree', als vereinfachter Programmablaufplan (PAP) dargestellt. Für die Dateiarbeit wurden Dateihandler verwendet, um die Dateien in einen geschützten 'read only'-Zugriffsmodus öffnen zu können, und im Falle eines Nichtfunktionierens Ausnahmefehler an den Nutzer auszugeben. Die Dateien werden mittels der Dateihandler zeilenweise gelesen, und Zeile für Zeile in einer temporären Stringliste, mit der Bezeichnung 'LReaderList', gespeichert. Das 'L' am Anfang des Namens der Stringliste steht für "local". Damit soll ausgedrückt werden, dass die Variable nur in dieser speziellen Prozedur benutzt wird, nicht aber im Rest der Unit. Auf jeden Eintrag (also jede Zeile) in der 'LReaderList' kann durch den Index der Liste zugegriffen werden. Durch die Nutzung der Tabulatoren als 'StrictDelimiter' (Trennzeichen), werden die 'LReaderList'-Einträge als Werte einer weiteren Stringliste gespeichert, welche als 'LWriterList' bezeichnet wurde. Aus dieser 'LWriterList' werden dann, durch Zugriff über die Listen-Indizes, die für die VirtualTreeview-Komponente wichtigen Daten in einen Array aus Containern geschrieben. Jedem Container werden die Chip-Attribute, eines Microarray-Experimentes, aus den Textdateien zugewiesen. Die zugehörige Container-Klasse wurde bereits im Abschnitt 3.3 genauer erklärt. Um die Chip-Attribute in den Spalten des virtuellen Baums anzuzeigen, wird auf die entsprechenden Container aus dem dynamischen Array zugegriffen, und diese werden schließlich zu den Knoten des VSTs sortiert.

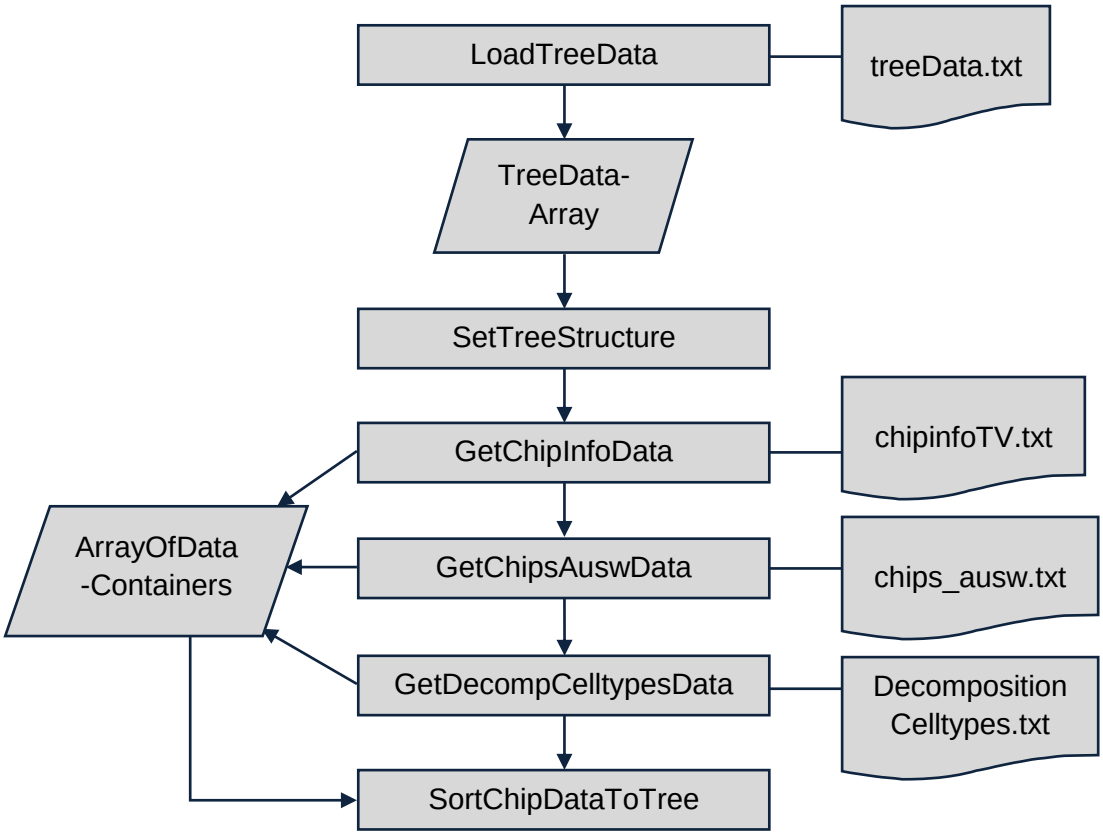


Abbildung 13 - PAP des 'OnClick'-Ereignisses zum Laden des virtuellen Baumes
Der 'TreeDataArray' wird genutzt, um die Struktur der VirtualTreeview zu erzeugen. Nachdem die Prozedur zur Erzeugung der Baumstruktur abgehandelt ist, werden die drei Prozeduren zum Laden der Chip-Attribute aufgerufen. Die Attribute werden per Dateihandler eingelesen und im 'ArrayOfDataContainers' gespeichert. Dieser Array ist dynamisch und global, wodurch seine Größe anpassbar ist, und der Zugriff von anderen Klassen aus erfolgen kann.

3.3 Anlegen einer Container-Klasse für Chip-Attribute

Für die neue Klasse wurde eine neue Unit, mit der Bezeichnung 'containerunit' angelegt. In dieser Unit erfolgte die Deklaration der Container-Klasse (Abbildung 14). Das Anlegen einer neuen Unit für die neue Klasse wäre kein Muss gewesen, ist aber unter Pascal- und Delphi-Programmierern üblich, um eine bessere Übersichtlichkeit des Quellcodes zu bewahren. Es wurden alle Attribute der Container-Klasse als 'private' deklariert, um sie vor Zugriffen von außerhalb der Klasse zu schützen. Die Attribute beginnen alle mit einem 'F', welches für "field" steht. Auch dies ist eine geläufige Gepflogenheit unter Delphi-Programmierern. Der Großteil der Attribute wurde als ShortString deklariert, da dieser 255 Zeichen lange Datentyp speicherschonender als der normale String ist. Einzig die Chip-Beschreibung wurde als String deklariert, da für diese 255 Zeichen nicht ausreichend sind. Ein Attribut ('FExprPeakPos') wurde als Integer definiert, welches den Index der Spalte in der Datei 'Decomposition Celltypes' angibt, in welcher sich der

höchste Expressionswert befindet. Nur mittels dieser Indizes ist es möglich, auf die entsprechenden Bezeichnungen der dominant exprimierten Zelltypen zu zugreifen.

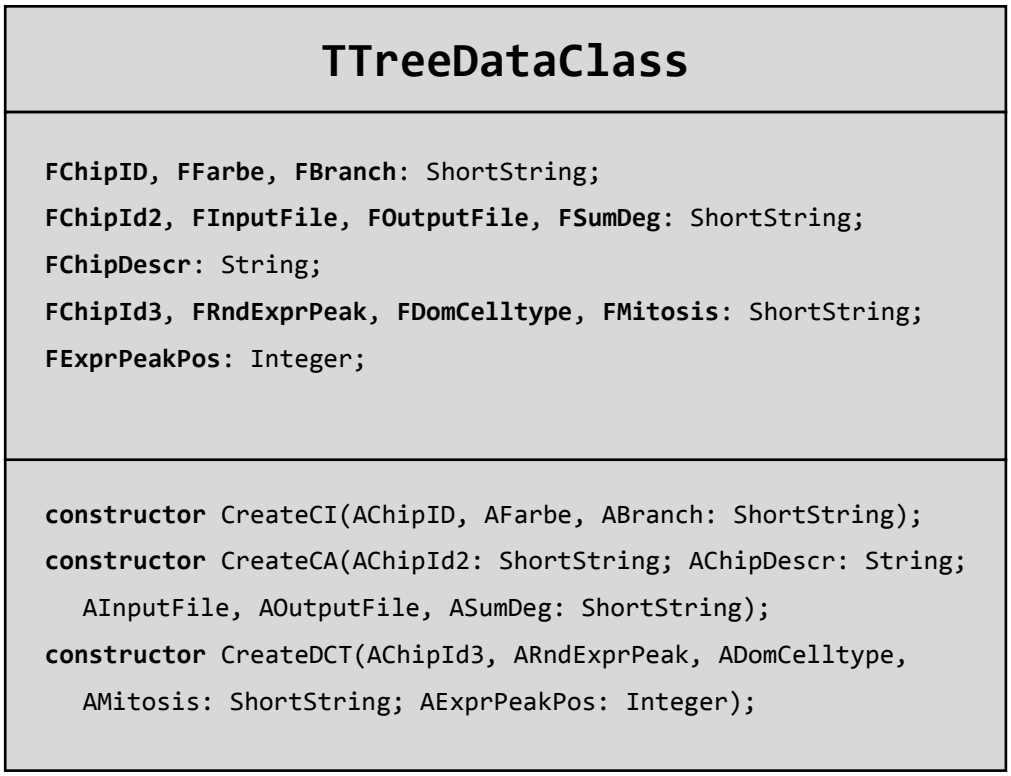


Abbildung 14 - Schema der Container-Klasse
Die Chip-Attribute wurden als 'private' und die Konstruktoren als 'public' definiert. Die Konstruktoren haben die Endung CI, CA und DCT, welches Akronyme sind, um die drei Konstruktoren der Dateien 'chipinfoTV', 'chips_ausw' und 'Decomposition Celltypes' zu unterscheiden.

Folglich wurde für jede Gruppe der Attribute, ein Konstruktor geschrieben. Die Attribute wurden nämlich gedanklich, je nach dem aus welcher Datei sie stammen, in drei Gruppen eingeteilt. Denn ein Konstruktor kann nur die Attribute aus einer Datei in einem Container abspeichern, die während des Programmablaufs gerade zur Verfügung stehen. Oder anders ausgedrückt, nur die Attribute, welche auch zu einem bestimmten Zeitpunkt während des Programmablaufs aus einer Datei mittels Handler ausgelesen werden, können genutzt werden, um mittels Konstruktor eine neue Instanz der Klasse zu erzeugen. Man hätte auch alle Attribute vorerst in einem Array aus Records speichern können, was aber eine zusätzliche Belastung des RAM-Speichers zur Folge gehabt hätte. Zum Schreiben und Lesen der Attribut-Felder mussten sogenannte 'properties' (Eigenschaften) definiert werden (Abbildung 15). Durch diese 'properties' kann dann von außerhalb der Klasse, der Zugriff auf die Attribut-Felder gesteuert werden, da sie ebenso wie die Konstruktoren als 'public' (öffentlich) deklariert wurden. Danach wurden

die Konstruktoren im Implementations-Teil der Klasse erzeugt, um die Attribute aus den Argumenten in die entsprechenden Attribut-Felder zu schreiben.

Properties (public)		
ChipID, **Farbe,** **Branch:** ShortString;	**ChipId2,** **InputFile,** **OutputFile,** **SumDeg:** ShortString; **ChipDescr:** String;	**ChipId3,** **RndExprPeak,** **DomCelltype,** **Mitosis:** ShortString; **ExprPeakPos:** Integer;

Abbildung 15 - Die 'properties' der Container-Klasse
Auf diese, als öffentlich deklarierten, Eigenschaften kann von außerhalb der Klasse zugegriffen werden. Dies geschieht mit Hilfe von Zeigern auf die entsprechende Objekt-Instanz der Container-Klasse.

Für die Darstellung der Knoten im VST wurde ein Record in der 'virtualtreeviwunit' angelegt (Abbildung 16). Die Records sind vom Programmierer eigens definierte Typen, welche von der Struktur her dem Datensatz einer Datenbank ähnlich sind. Diese Typen sind, im Gegensatz zur Datenbank, allerdings nur zur Laufzeit vorhanden. Records dienen dazu unterschiedliche, aber logisch zusammen gehörige Daten zusammenzufassen [URL-16]. Der Record besteht aus den Variablen 'FObject' vom Typ der neuen Container-Klasse 'TTreeDataClass', 'FCaption' vom Typ String, und einer Variable 'FGroup' vom Typ Integer. Über das Feld 'FObject' wird der Zugriff auf alle Attribute der Klasse 'TTreeDataClass' gesteuert. So kann im weiteren Programmverlauf bspw. mit dem Befehl 'Data^.FObject.ChipID' - ein Zeiger auf eine Objektinstanz - auf das Feld-Attribut 'FChipID' der Klasse 'TTreeData' zugegriffen werden, wenn 'Data' vom Typ 'PTreeData' ist. Das 'P' in 'PTreeData' steht für 'Pointer' und bedeutet, dass es sich um eine Zeigervariable auf den Speicherbereich eines bestimmten Records handelt. Die Zeigervariable enthält selbst keine Daten, sondern nur die Speicheradresse der Daten [BINZINGER (2006), S. 240].

```
type

  (* pointer on record with object *)
  // the variable FObject of the record will later contain...
  // ...the data of the declared object
  PTreeData = ^TTreeData;
  TTreeData = record
    FObject : TTreeDataClass;
    FCaption: String;
    FGroup: Integer;
  end;
```

Abbildung 16 - Deklaration des Records 'TTreeData'
Der Record beinhaltet die Felder: 'FObject', 'FCaption' und 'FGroup'. 'FObject' steuert den Zugriff auf eine Objekt-Instanz der Container-Klasse. Das Feld 'FCaption' wurde angelegt, um ihm die 'chip description' zuzuweisen. Das Feld 'FGroup' wurde als Variable zur Markierung der Chips angelegt, um deren Gruppenzugehörigkeit (A oder B) festzuhalten.

Der Eintrag 'FCaption' ist für das Anlegen der Baumstruktur in der ersten Spalte von Bedeutung, um die Einträge aus der Datei 'treeData.txt' darzustellen. Die Variable 'FGroup' des Records wurde als "Marker" für die Chip-Listen A und B deklariert. Sie ist vom Typ Integer, also einer natürlichen positiven Zahl, der entweder der Wert 1 oder 2 zugewiesen wird. Eine 1 wird dem Integer 'FGroup' zugeteilt, wenn der Chip der Gruppe A zugeordnet wird. Im Gegensatz dazu wird der Variable 'FGroup' der Wert 2 zugewiesen, wenn der Chip der Gruppe B zugewiesen wird. Dieser "Marker" wird ebenfalls verwendet, um die Chip-Listen abzuspeichern oder um diese aus einer vorhandenen Datei zu laden.

3.4 Erstellen eines erweiterten Kontrollpanels

Das Kontrollpanel auf der Registerkarte 'Virtual Treeview' ist die wichtigste Komponente zur Steuerung des VSTs. Durch die grafischen Komponenten aus der VCL von Lazarus, wie z. B. Buttons und Schiebereglern, lassen sich die Prozeduren zur Initialisierung und zur interaktiven Bearbeitung des virtuellen Baums auslösen (Abbildung 17 und 18). Die Prozeduren laufen dabei aber immer versteckt im Hintergrund des GUI ab, und sind somit vor dem Anwender verborgen. Der Benutzer der Software setzt aber durch seine Aktionen auf dem Kontrollpanel des GUI die entsprechenden Prozeduren des Programms in Gang.

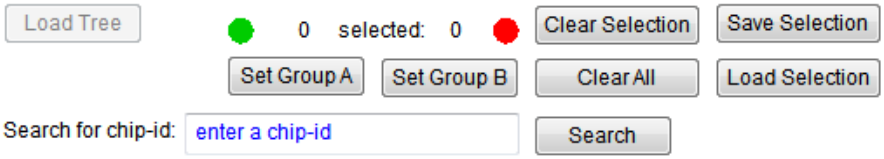


Abbildung 17 - linke Seite des Kontrollpanels
Zu erkennen ist der (inaktivierte) Button zum Laden des Baumes, ebenso wie die Buttons zum Hinzufügen der Knoten zur Gruppe A oder B. Beim Label 'selected:' soll angezeigt werden, wie viele Knoten bereits zur Gruppe A oder B gehören. Weiterhin sind die Buttons zum Löschen der markierten Auswahl ('Clear Selection'), sowie der gesamten Auswahl zu sehen. Mit 'Save Selection' und 'Load Selection' sollen die Chip-Listen gespeichert, bzw. geladen werden. Im unteren Bereich ist ein Suchfeld angelegt, welches zum Auffinden eines bestimmten Chips eingesetzt werden soll.

Es wurde darauf geachtet das Kontrollpanel möglichst logisch zu ordnen. So geschehen, indem die Komponenten von links nach rechts auf dem Panel so angeordnet wurden, wie sie auch nacheinander zu bedienen sind. Also zu allererst soll das Laden des VST erfolgen. Durch Klicken auf den Button 'Load Tree' wird im Hintergrund des Programms das OnClick-Ereignis ausgelöst, welches in Abschnitt 3.2 beschrieben wurde. Für die Initialisierung der Chip-Attribute wurde ein Ladebalken unter dem Button 'Load Tree' eingebunden, der den aktuellen Stand des Ladevorgangs anzeigt. Dieser Ladebalken ist aber auf Abbildung 17 nicht sichtbar, da er nur während der Initialisierung der Chip-Attribute angezeigt werden soll.

Weiterhin sind rechts neben dem Button 'Load Tree' die Buttons zur Auswahl der Chip-Listen erstellt worden. Die Chips sollen mit Hilfe der Buttons 'Set Group A' und 'Set Group B' zu einer der zwei Gruppen hinzugefügt werden. Dabei wurde dem Button 'Set Group A' die Prozedur 'BtGroupAClick', und dem Button 'Set Group B' die Prozedur 'BtGroupBClick' zugewiesen. Jede dieser Prozeduren durchläuft jeweils den gesamten VST und weist allen selektierten Knoten die entsprechende Gruppe zu. Dabei dürfen aber nur diese Knoten zur entsprechenden Gruppe hinzugefügt werden, die Chip-Attribute besitzen. Also bspw. keine Elternknoten oder leere Knoten. Die Gruppenzuweisung erfolgt unter Zuhilfenahme einer Integer-Variable 'FGroup', die für jeden Knoten gesetzt werden kann. Sie erhält den Wert 1 für die Gruppe A und den Wert 2 für die Gruppe B. Durch die Prozedur 'VSTGetImageIndex', wird die Variable 'FGroup' jedes Knotens überprüft. Hat sie den Wert 1, wird vor den Knoten in der VirtualTreeview ein grüner Punkt angezeigt. Hat sie hingegen den Wert 2, so wird ein roter Punkt aus der Imagelist angezeigt.

Um die Auswahl auch rückgängig machen zu können, wurden die zwei Buttons 'Clear Selection' und 'Clear All' angelegt. Wie die Bezeichnung schon verrät, handelt es sich

bei dem Button 'Clear Selection' um eine Schaltfläche, durch deren Auslösung die aktuelle Auswahl von Knoten, die zur Gruppe A oder B gehören, durch den Benutzer gelöscht werden kann. Dafür wird eine Prozedur ausgelöst, welche den gesamten VST nach allen selektierten Knoten durchsucht, und deren Variable 'FGroup' auf den Wert 0 setzt. Mit dem Auslösen des Buttons 'Clear All', wird die Gruppenzugehörigkeit von allen Knoten des virtuellen Baums aufgehoben.

Für das Suchen nach speziellen Chip-Einträgen wurde ein Label mit der Bezeichnung 'Search for chip-id: ', ein Textfeld vom Typ TEdit zur Eingabe der gesuchten Chip-ID, sowie ein Button zum Auslösen der Suchfunktion erstellt. Mit Hilfe dieser Suchfunktion können bestimmte Microarray-Experimente anhand der Chip-ID im virtuellen Baum gefunden werden. Dabei wird, nach Klicken auf den Button 'Search', der Fokus der VirtualTreeview auf den entsprechenden Knoten gesetzt und der gesuchte Eintrag markiert. Die zugehörige Prozedur im Hintergrund wurde so geschrieben, dass sie alle Knoten des VSTs durchläuft, und von jedem Knoten das Attribut 'ChipID' mit dem im Textfeld eingegebenen String vergleicht. Durch eine spezielle Funktion ist das Eingabefeld nicht "case sensitive", wodurch es egal ist, ob die Buchstaben vom Benutzer im Suchfeld als Groß- oder Kleinbuchstaben eingegeben werden.

Die letzten beiden Buttons, die auf der linken Seite des Kontrollpanels angelegt wurden, sind die Buttons 'Save Selection' und 'Load Selection'. Durch Betätigen des Buttons 'Save Selection' wird eine Prozedur ausgelöst, die zuerst einen Speicherdialog öffnet. In diesem Speicherdialog legt der Anwender eine Datei in einem beliebigen Ordner fest, in der die Knoten, welche einer Gruppe zugehören, abgespeichert werden. Dann werden durch die Prozedur 'BtSaveSelectionClick' das Attribut 'FChipID', sowie die Integer-Variable 'FGroup', jedes in den Gruppen A und B gelisteten Knotens, zeilenweise in diese neu angelegte Datei geschrieben. Dies geschieht mittels eines Dateihandlers im 'write only'-Modus.

Das Laden von vorher bereits abgespeicherten Chip-Listen, wurde durch eine Prozedur realisiert, die durch Klicken auf den Button 'Load Selection' startet. Auch hier wird vorerst ein Dialog geöffnet, in dem der Nutzer eine zuvor abgespeicherte Datei auswählen kann. Das Laden aus den Dateien erfolgt, ebenso wie beim Speichern, mit Hilfe eines Dateihandlers, hier allerdings im 'read only'-Modus. Nach dem Laden werden die entsprechenden Knoten durch die Ausführung der Prozedur 'VSTGetImageIndex', mit den dazugehörigen grünen oder roten Punkten aus der Imagelist markiert.

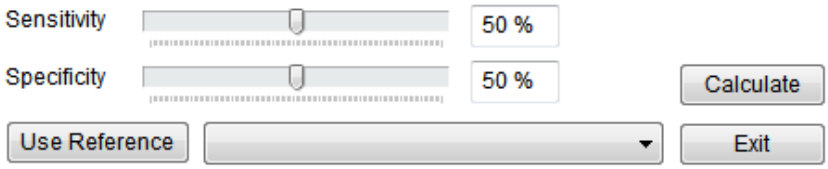


Abbildung 18 - rechte Seite des Kontrollpanels
Zu sehen sind zwei Schieberegler zum Einstellen der Sensibilität und der Spezifität. Außerdem ein Button und eine Kombinationsbox zur Auswahl eines Zelltyps, der als Bezugswert genutzt werden kann. Ganz rechts befindet sich der Button zum Berechnen der Gen-Listen, und ein Button zum Verlassen der Programmkomponente 'Differential Expression'.

Danach wurden auf der rechten Seite des Kontrollpanels grafische Komponenten zum Festlegen spezieller Parameter erzeugt. Festgelegt werden können die Werte für die Sensibilität und die Spezifität, die in einem Wertebereich von 0 - 100 % liegen. Dafür wurden aus der Vorgängerversion der 'Differential Expression'-Komponente, die beiden Schieberegler mit ihren Bezeichnungen 'Sensitivity' und 'Specificity' übernommen. Folglich soll die Auswahl eines Zelltyps, der als Referenz dient, möglich sein. Dazu wurde eine Combobox erstellt, welche eine Liste von Gewebetypen enthält. Diese Liste kann für die Berechnung genutzt werden, nachdem der Benutzer den Button 'Use Reference' aktiviert hat. Diese Referenzen sind Gewebetypen, welche durch die Prozedur zum Laden der Datei 'Decomposition Celltypes' aus dieser Datei entnommen werden. Dies geschieht während der Initialisierung der VST, nach dem Auslösen des Buttons 'Load Tree'.

Durch das Klicken auf 'Calculate' wird eine Prozedur ausgelöst, die alle vorher vom Benutzer eingestellten Parameter berücksichtigt. Die dafür notwendige Prozedur konnte aus dem Quelltext der alten Version der 'Differential Expression'-Komponente übernommen werden. Die berechneten Gen-Listen werden dann in zwei Tabellen des Typs TStringGrid, in der zweiten Registerkarte 'Gene Lists' angezeigt. Die Eigenschaften der Tabellen wurden so verändert, dass die Spalten vergrößert oder verkleinert werden können.

4 Ergebnisse und Diskussion

Alle wesentlichen Anforderungen, welche an die VirtualTreeview-Komponente gestellt wurden, konnten im gegebenen Zeitraum realisiert werden. Zu den Anforderungen gehörte auch, dass die neue Version der 'Differential Expression'-Komponente, im Gegensatz zur alten Version, für den 64-bit-Betrieb programmiert werden musste. Um eine 64-bit-Version der 'Differential Expression'-Moduls zu erstellen, wurde deshalb von der IDE Lazarus eine 64-bit-Version benutzt. Im 32-bit-Modus des XpressionXplorers konnten nämlich nur maximal etwa 3.900 Chips adressiert werden. Dies wäre für die Auswertung von über 11.000 Chip-Datensätzen, die zum Zeitpunkt des Projekts bereits existierten, nicht ausreichend gewesen. Daher wurde die neue Programmkomponente im 64-bit-Modus programmiert .

Bei der Erzeugung des Formulars für die neue 'Differential Expression'-Komponente wurde darauf geachtet, dass dieses einen Wiedererkennungswert zum Formular der älteren Programmkomponente besitzt. Dadurch soll den Anwendern der Umstieg, zur jetzigen 'Differential Expression'-Version, erleichtert werden. Allerdings musste für das erweiterte Formular zusätzlich eine Registerkartenkomponente verwendet werden. Eine Registerkarte wurde für die Bedienung der VirtualTreeview erzeugt und eine weitere für die Darstellung der Gen-Listen. Denn mit der Benutzung der VirtualTreeview-Komponente sind acht zusätzliche Spalten zur Darstellung der Chip-Attribute im VST vorhanden, wodurch wesentlich mehr Platz auf dem Formular benötigt wird, als mit der zuvor verwendeten Treeview. Da das Kontrollpanel um einige Buttons erweitert wurde, musste dem Panel auf dem Formular des überarbeiteten Programmmoduls ebenfalls mehr Platz eingeräumt werden, als in der Vorgängerversion der Programmkomponente.

4.1 Übersicht der Programmerweiterungen

Das VirtualTreeview-Package hat einen größeren Funktionsumfang als das einfacher gehaltene, vorher benutzte, Treeview-Package. So ermöglicht das ViertualTreeview-Package bspw., das Anlegen mehrerer Spalten im VST. Somit konnten die biologischen und technischen Chip-Attribute aus den Textdateien 'chipinfoTV', 'chips_ausw' und 'Decomposition Celltypes', spaltenweise im VST eingebunden werden. Das neue

'Differential Expression'-Programmodul wurde, neben dem Einbinden des VirtualTreeview-Packages, um folgende Funktionen erweitert:

- Die Zuordnung von Knoten aus dem VST zu den zwei Gruppen A und B kann nun auch wahlweise über das Pop-up-Menü erfolgen.
- Das neue Programmmodul bietet jetzt die Möglichkeit des Abspeicherns und Ladens der Chip-Listen in, bzw. aus Textdateien.
- Eine weitere Verbesserung ist die integrierte Suchfunktion, mit der man nach einer bestimmten Chip-ID im VST suchen kann.

Folgende Funktionen sind, zur Erhöhung der Anwenderfreundlichkeit, zusätzlich in die neue 'Differential Expression'-Programmkomponente des XpressionXplorers integriert worden:

- Es gibt zwei weitere Einträge im Pop-up-Menü, um die Baumstruktur des VSTs vollständig aus- oder einzuklappen, so dass alle oder gar keine Knoten - außer der Root-Knoten - zu sehen sind.
- Ein Ladebalken erscheint während der Initialisierung des VSTs, der dem Anwender den aktuellen Stand der Initialisierung der Chip-Attribute anzeigt.
- Die Tabellenspalten mit den Gen-Listen können in der neuen Version vergrößert oder verkleinert werden, um die Einträge komplett sichtbar zu machen.
- Eine Prozedur zum alphabetischen Ordnen der Knoten wurde implementiert. Diese wurde jedoch auskommentiert, kann aber bei Bedarf nachträglich eingebunden werden.

Ein Großteil der hier vorgestellten Programmerweiterungen, der erneuerten 'Differential Expression'-Programmkomponente des XpressionXplorers, befinden sich in Form von Screenshots im Anhang.

4.2 Die Initialisierung der Chip-Attribute

Die Initialisierung der Chip-Attribute erfolgt, nach Aufrufen des 'Differential Expression'-Formulars, durch Klicken auf den Button 'Load Tree', und dauert einige Zeit, bis sie komplett abgeschlossen ist. Dabei wird das in Abschnitt 3.2 beschriebene 'OnClick'-Ereignis ausgelöst. Das Ereignis bewirkt, dass zuerst die Baumstruktur aus der Datei 'treeData.txt' mit allen Knoten im VST gezeichnet wird. Die Knotenbezeichnungen entsprechen den Gewebearten, aus denen die Proben für die Microarray-Experimente entnommen wurden. Danach beginnt die eigentliche Initialisierung, indem alle Chip-

Datensätze in einen dynamischen Array aus Containern geladen werden. Dadurch beinhaltet folglich jeder Container einen Microarray-Datensatz. Die technischen Chip-Attribute 'farbe' und 'branch' werden dazu aus der Datei 'chipinfoTV.txt' ausgelesen. Ebenso werden die technischen Chip-Attribute 'chip-id', 'chip description', 'Eingabe-Datei', 'Ausgabe-Datei' und 'Summe Degradation ACTB' aus der Textdatei 'chips_ausw' initialisiert. Des weiteren werden aus der Datei 'Decomposition Celltypes.txt' die biologischen Attribute 'dominant celltype', 'expression factor' und 'virtual mitotic index' geladen. Die Zuordnung der richtigen Chip-Attribute zu einem Datensatz erfolgt mittels der Chip-IDs, die für jedes Microarray-Experiment einmalig sind und in jeder der genannten Dateien vorkommen. Initialisiert werden aber nur die Chip-Attribute zu den zuvor im Startfenster der XpressionXplorers geladen *_mp.txt-Dateien. Kurz gesagt, wird aus jedem Microarray-Experiment, das in den vorher geladenen *_mp.txt-Dateien gespeichert ist, ein kompletter Datensatz von Chip-Attributen generiert. Diese komplettierten Datensätze - programmintern in Form von Containern (siehe Abschnitt 3.3) - werden den zugehörigen Zweigen, bzw. Knoten des Baums zugeordnet. In der VST-Komponente werden die unterschiedlichen biologischen und technischen Daten dann zu den neun Spalten der VirtualTreeview geordnet, und somit für den Anwender übersichtlich dargestellt.

Einen entscheidenden Nachteil hat die erneuerte 'Differential Expression'-Komponente allerdings gegenüber der Vorgängerversion. Für die Initialisierung der gesamten Chip-Attribute wird eine Gesamtzeit von ca. 1½ Minuten benötigt. Denn alle Chip-Attribute, die aus den Dateien geladen werden, müssen durch einen Algorithmus zueinander sortiert werden. Als Algorithmus, der diese Sortierung übernimmt, wurden einfachheitshalber zwei geschachtelte FOR-Schleifen benutzt, welche die Chip-IDs gegeneinander vergleichen. Dies ist nicht die optimale Lösung. Die Wartezeit ist aber unabhängig von der Anzahl der verwendeten *_mp.txt-Dateien und wird dem Anwender durch die Anzeige eines Ladebalkens signalisiert. An diesem Punkt könnte eine Optimierung durch das Einbinden eines besseren Sortieralgorithmus'- wie z. B. Quicksort - erfolgen, der die Attribute anhand der Primärschlüssel (Chip-IDs) schneller einander zuordnet.

4.3 Die Vorteile der VirtualTreeview

Das Verwenden einer VirtualTreeview in der neuen 'Differential Expression'-Programmkomponente, hat einige entscheidende Vorteile gegenüber der bis dahin verwendeten einfachen Treeview. Bspw. erfolgt eine Aktualisierung des VSTs rasend schnell, so dass der Benutzer davon nichts mitbekommt, wenn er z. B. die Knoten zu

einer Gruppe hinzufügt und zeitgleich die grünen oder roten Markierungen vor den Knoten angezeigt werden. Oder wenn der Nutzer nach einer Chip-ID sucht und sofort nach Klicken des Buttons 'Search' der gesuchte Knoten im VST fokussiert wird. Ein zusätzlicher Vorteil - den die Verwendung der VirtualTreeview mit sich bringt - ist, dass die Auswahl der Knoten unter Verwendung der Tasten SHIFT und STRG auf der Tastatur erfolgen kann, oder durch Aufziehen eines Auswahlrechtecks mit der Maus. Dadurch ist es möglich mehrere Knoten gleichzeitig auszuwählen, um sie danach zur Gruppe A oder B hinzuzufügen.

Der wohl größte Vorteil, den die Benutzung der VST-Komponente mit sich bringt, ist das statt nur einer, nun mehrere Spalten dargestellt werden können. Somit ist die Darstellung der verschiedenen biologischen und technischen Informationen (Chip-Attribute) zu den Microarray-Experimenten in den verschiedenen Spalten erst möglich geworden. Diese Chip-Attribute dienen als Auswahlkriterien für den geübten Benutzer, um ihm die Auswahl für die Chip-Listen zu erleichtern. Dabei sind für den Nutzer vor allem die biologischen Chip-Attribute von Bedeutung, die in den grün hinterlegten Spalten 2 bis 4 angezeigt werden. Darin wird z. B. der dominante Zelltyp, mit dem höchsten Expressionsfaktor und der Wert, der den aktuellen Grad der Mitose angibt (Zellteilungsfaktor), abgebildet.

Statt die reinen Expressionswerte anzuzeigen, könnte man diese auch mit Hilfe von Prozentbalken noch anschaulicher gestalten. Um dies zu realisieren, wären aber noch zusätzliche Recherchen in den Demos des VirtualTreeview-Packages nötig gewesen. Zusätzlich wäre es möglich eine Bearbeitung des Attributs 'farbe', direkt in der entsprechenden Spalte des VSTs vom Benutzer vornehmen zu lassen. Dafür könnte z. B. ein Auswahlfeld nach dem Klicken auf den entsprechenden 'farbe'-Eintrag aufklappen, um eine andere Farbzuordnung selektieren zu können. Um dies umzusetzen, müsste aber auf die Datei 'chipinfoTV.txt' ein Vollzugriff erfolgen, um den Eintrag des Attributs dort ebenfalls zu ändern. Die Farbzuordnungen werden im späteren Programmverlauf des XpressionXplorers benötigt, um die Chips in einem 'Scatterplots' farbig darzustellen.

4.4 Neuer Funktionsumfang der 'Differential Expression'

Wie auch bei der alten Version, so ist die Zuordnung der Knoten des VSTs zu den Gruppen A und B von entscheidender Bedeutung, um die Chip-Listen zu generieren. Damit die Auswahl für den Nutzer erkenntlich bleibt, werden die Markierungspunkte

genau wie bei der einfachen Treeview, direkt vor der Knotenbezeichnung ('chip description') angezeigt. Aber im Gegensatz zur alten Variante, wurden statt der Farben Blau und Orange, die Farben Grün und Rot verwendet für eine noch deutlichere Kennzeichnung. Diese können in der Imagelist angepasst werden, um bei Bedarf die ursprünglichen oder andere Markierungsfarben zu verwenden. Dies wäre, z. B. bei Benutzer mit einer Rot-Grün-Schwäche, notwendig.

Die neue Version der 'Differential Expression'-Komponente bietet die Möglichkeit, die Auswahllisten abzuspeichern oder bereits abgesicherte zu laden (Abbildung 24 im Anhang). Dies hat den großen Vorteil, dass der Anwender nicht bei jeder Ausführung des XpressionXplorers die Listen manuell komplett von Neuem zusammenstellen muss. Dadurch gelang eine enorme Steigerung der Effizienz beim Arbeiten mit der 'Differential Expression'-Komponente. Außerdem hat dies den Vorteil, dass die Auswahllisten nun auch unter verschiedenen Benutzern ausgetauscht oder archiviert werden können. Damit ist es möglich geworden, die Arbeiten zu einem späteren Zeitpunkt fortzusetzen oder für anderen Nutzer nachvollziehbar zu machen. Mit dieser Neuerung wird es den Genexpressions-Forschern also ermöglicht, die Chip-Listen abzuspeichern, welche gute Resultate beim Suchen von markanten Genen liefern. Somit sind die Genexpressions-Analysen nun reproduzierbar. Das Speichern der Chip-Listen geschieht mittels Textdateien, in welche die String-Variable 'FChipID' und die Integer-Variable 'FGroup' tabulatorgetrennt geschrieben werden.

Eine Möglichkeit der Verbesserung wäre es, die Listen als XML-Dateien abzuspeichern und dabei die komplette Baumstruktur abzusichern. Denn wenn ein Benutzer die Chip-Listen abspeichert, aber der nächste Benutzer andere *_mp.txt-Dateien einliest, werden die fehlenden Einträge nicht mit geladen. Wenn nicht die gesamte Baumstruktur mitgespeichert werden soll, wäre das Erstellen eines Hinweises auf fehlende Einträge im Baum nach dem Laden hilfreich, welcher bisher nicht existiert. Um dies noch umzusetzen, bedarf es aber keinem großen Programmieraufwand. Es müsste lediglich geprüft werden, welche Chip-IDs in der Textdatei vorkommen, aber nicht im gegenwärtig initialisierten VST.

Weiterhin ist die Möglichkeit des Durchsuchens der VirtualTreeview nach einem bestimmten Microarray-Experiment anhand der Chip-ID, in der neuen Version der 'Differential Expression'-Komponente, umgesetzt worden (Abbildung 25 im Anhang). Dies war bei der ursprünglichen 'Differential Expression'-Komponente nicht möglich. Dort mussten die Benutzer mühsam den gesamten Baum manuell durchsehen um einen bestimmten Eintrag zu finden. Die neue Suchfunktion übernimmt diese Durchsuchung

des VSTs automatisch und sehr schnell, so dass der Fokus im Bruchteil einer Sekunde, auf den gesuchten Eintrag gerichtet wird. Gleichzeitig wird der zugehörige Knoten in der VirtualTreeview markiert.

Eine Verbesserung der Suchfunktion könnte dahingehend erfolgen, dass eine Chip-ID in der Suchmaske nicht erst vollständig eingegeben werden muss. Denn aktuell ist es notwendig die Chip-ID, zwar nicht case-sensitive, aber sonst bis auf das letzte Zeichen genau einzugeben, damit der Eintrag gefunden wird. Zur Optimierung der Suchfunktion, könnte diese mit einer Autovervollständigung versehen werden. Dadurch würde sich nach Eingabe jedes einzelnen Zeichens, in einer Anzeige die Anzahl der möglichen Chip-Einträge immer mehr verringert, bis schließlich der gesuchte Eintrag gefunden wird.

Eine Prozedur zum alphabetisch aufsteigenden oder absteigenden Ordnen der Baumstruktur und der entsprechend zugeordneten Knoten kann implementiert werden. Diese wurde schon geschrieben aber auskommentiert, und der Ansatz damit nicht weiter verfolgt. Die Prozedur könnte aber benutzt werden, um bspw. durch zwei weitere Einträge im Pop-up-Menü oder durch Klicken auf den Spaltenheader der ersten Spalte des VSTs eine alphabetische Sortierung aller Knoten durchführen zu lassen.

4.5 Parameterauswahl und Berechnung von Gen-Listen

Die Schieberegler zum Einstellen der Sensitivität und Spezifität wurden aus der Vorgängerversion übernommen. Ebenso die Combobox zu Auswahl eines Gewebetyps, der als Referenz genutzt werden kann, für die Berechnung der Gen-Listen. Außerdem ein Button vom Typ TToggleBox mit der Bezeichnung 'Use Reference', der zur Verwendung eines Gewebetyps vom Anwender aktiviert werden muss. Die Prozeduren, die im Hintergrund des GUI ablaufen, um die Parameter einzustellen, wurden aus der Vorgängerversion der 'Differential Expression'-Komponente weitestgehend übernommen. Ebenso die Prozedur, welche nach Betätigen des Buttons 'Calculate' startet und die Gen-Listen in der Registerkartenkomponente 'Gene Lists' berechnet.

Die Berechnung wird vom Benutzer ausgeführt, um mögliche Kandidaten-Gene zu finden. Diese Gene können als spezifische Biomarker für Krankheiten oder andere Veränderungen in bestimmten Gewebearten nützlich sein. In den Tabellen der Registerkarte 'Gene Lists' sollen die Anwender der Analyse-Software schließlich die besten potentiellen Marker-Gene erkennen können, um Biomarker für diagnostische Zwecke aufzuspüren. Dafür wird allerdings ein entsprechendes Know-how von den

Anwendern vorausgesetzt. Die Listen werden wie bei der alten Version der Programmkomponente in Tabellenform mittels der LCL-Komponente TStringGrid, in der Registerkarte 'Gene Lists', erzeugt. Das TStringGrid ist ein Gitter, das mit Zeichenketten gefüllt werden kann [CANNEYT (2011), S. 418]. In diesen Gittern sind letzten Endes die ersten 1999 Probesets mit den Genen, welche die höchsten errechneten Werte haben, absteigend aufgelistet (Abbildung 28 im Anhang). Die Spalten können nun, nach erfolgtem Anpassen der TStringGrid-Eigenschaften, vergrößert werden, um z. B. die Bezeichnungen der Gene vollständig anzusehen. Trotzdem sind die TStringGrid-Komponenten nicht optimal zur Darstellung der Listen, da ihre gesamte Breite nicht anpassbar ist. Somit müssen die horizontalen Scrollbars benutzt werden, um durch die Gitter zu navigieren. Eine Möglichkeit wäre es die Gitter untereinander darzustellen, worauf aber aufgrund der besseren Übersichtlichkeit verzichtet wurde.

Der Wert 'value' in den letzten Spalten der Gitter bezeichnen das Ergebnis des Vergleichs der Expressionswerte je Probeset, für Chips die in den Chip-Listen zusammengefasst worden sind. Es gibt für jedes Probeset und jede Liste (A oder B) einen berechneten 'value'. Desto größer dieser Wert ist, desto höher ist die Wahrscheinlichkeit, dass es sich bei der auf dem Probeset hybridisierten mRNA, und des damit korrelierenden Gens, um einen Biomarker handelt. Spezifische Biomarker, z. B. für eine Krankheit, können durch die Gegenüberstellung von gesundem und krankhaft verändertem Gewebe ermittelt werden. Ebenso können Biomarker für ein, durch andere Einflüsse verändertes, Gewebe entdeckt werden. So kann sich z. B. das Genexpressionsprofil von Zellen, aus bestimmten Gewebeproben von Patienten, nach längerer Medikamenteneinnahme verändern. Die dabei gefundenen Biomarker können dann als Anzeiger für die Nebenwirkungen dieser Medikamente fungieren. Dieses Wissen kann u.a. für die Entwicklung individueller Medikamente von Bedeutung sein. Biomarker können auch solche Gene sein, die aufgrund unterschiedlicher Lebensfaktoren (z. B. Alter, Geschlecht, Lebensweise) unterschiedlich stark exprimiert werden und dann ebenfalls, durch einen hohen berechneten Expressionswert ('value'), in den Gen-Listen auffallen.

4.6 Zusammenfassung

Die Ersetzung der Treeview durch das innovativere VirtualTreeview-Package, in der erneuerten 'Differential Expression'-Komponente des XpressionXplorers, verlief erfolgreich. Durch die Nutzung der VirtualTreeview können nun alle Chip-Attribute, welche aus den Genexpressionsdatenbanken stammen, in der VST-Komponente

anschaulich, spaltenweise angezeigt. Diese neue Ansicht stellt somit, im Gegensatz zur alten Version, die Auswahlkriterien zum Erstellen der Chip-Listen ausführlich dar. Weiterhin wurde die Programmkomponente dahingehend erweitert, dass diese Chip-Listen jetzt abgespeichert und auch wieder geladen werden können. Außerdem wurde eine Suchmaske eingebaut, mit deren Hilfe nach einem bestimmten Chip im VST gesucht werden kann. Das Pop-up-Menü zum gruppieren der Chips wurde, als optionale Anforderung, ebenfalls realisiert. Zur Verbesserung der Anwenderfreundlichkeit wurden noch kleine, zusätzliche Ergänzungen verwirklicht. So kann die virtuelle Baumansicht bspw. nun, durch bestimmte Einträge im Pop-up-Menü, komplett ein- oder ausgeklappt werden.

4.7 Ausblick

In der Projektplanung ist es ferner vorgesehen, die berechneten Gen-Listen, nach der Berechnung, auf 2D-Objekte kartieren (mappen) zu lassen. Für das Mappen der Gen-Listen würde sich z. B. der menschliche Chromosomensatz sehr gut eignen. Ein Programm, welches dies bereits realisieren kann, heißt 'Caryoskop', welches allerdings in Java geschrieben wurde. Diesem Programm können beim Aufrufen Parameter übergeben werden. Eine Möglichkeit, damit das gesamte Programm nicht erst in Object Pascal portiert werden muss, wäre es einen skriptorientierten Programmieransatz zu wählen. Damit ist es denkbar, dass das 'Caryoskop' extern unter der Übergabe von Parametern aus dem XpressionXplorer heraus gestartet wird, und dann die Gen-Listen auf den Chromosomenkarten gezeichnet werden. Eine weitere Möglichkeit, die zukünftig angestrebt werden soll, ist das Mappen auf metabolische Pathways, auf denen die Gen-Listen dann ebenfalls abgebildet werden könnten. Dazu existieren auch schon einige Programme, die dafür in den XpressionXplorer integriert werden müssten.

Literatur

Fachbücher

[BINZINGER (2006)]
Binzinger, Thomas (2006): Jetzt lerne ich Delphi. 1. Auflage: Markt+Technik Verlag München

[CANNEYT (2011)]
Van Canneyt, Michaël; Gärtner, Mattias; Heinig, Swen; Monteiro de Carvalho, Felipe; Ouedraogo, Inoussa; Weustink, Marc; Braun, Jörn (2011) : Lazarus: Klassenbibliothek und DIE. 2. Auflage: Computer & Literatur Verlag Böblingen

[LÖFFLER (2008)]
Löffler, Georg (2008): Basiswissen Biochemie mit Pathobio-chemie. 7. Auflage: Springer Medizin Verlag Heidelberg

[MADIGAN (2006)]
Madigan, Michael; Martinko, John (2006): Brock Mikrobiologie. 11. Auflage: Pearson Studium München

[MÜLHARDT (2009)]
Mülhardt, Cornel (2009): Der Experimentator – Molekularbiologie/ Genomics. 6. Auflage: Spektrum Akademischer Verlag Heidelberg

[MÜLLER (2004)]
Müller, Hans-Joachim; Röder, Martin (2004): Der Experimentator – Microarrays. 1. Auflage: Elsevier Spektrum Akademischer Verlag München

[SELZER (2004)]
Selzer, Paul M.; Marhöfer, Richard J.; Rohwer, Andreas (2004): Angewandte Bioinformatik: Eine Einführung. 1. Auflage: Springer-Verlag Berlin Heidelberg

Zeitschriften

[HOFFMANN (2006)]
F. Hoffmann-La Roche AG(2006): Gene und Gesundheit. 2. Auflage: Corporate Communications Basel

[HÜSING (2012)]
Hüsing, Bärbel (2012); Individualisierte Medizin – Potenziale und Handlungsbedarf. Zeitschrift für Evidenz, Fortbildung und Qualität (ZEFQ) 104: Fraunhofer ISI Karlsruhe

Internetadressen

[URL-1]
Freyer, Timo: Molekulare Medizin, http://flexikon.doccheck.com/de/Molekulare_Medizin, verfügbar am 25.06.2013

[URL-2]
http://de.wikipedia.org/wiki/Genom#Eukaryoten, verfügbar am 30.07.2013

[URL-3]
http://www.nature.com/nature/journal/v491/n7422/full/nature11632.html, verfügbar am 30.07.2013

[URL-4]
http://www.affymetrix.com/estore/browse/products.jsp?productId=131455#1_1, verfügbar am 11.08.2013

[URL-5]
http://bme240.eng.uci.edu/students/06s/fayers/fabrication.htm, verfügbar am 11.09.2013

[URL-6]
http://www.uni-saarland.de/fak8/heinzle/de/teaching/Aufbaupraktikum_Bioinf/V5_Meese_Aufbau_bioinfo_2008.pdf, verfügbar am 11.08.2013

[URL-7]
http://bitesizebio.com/articles/introduction-to-dna-microarrays/, verfügbar am 11.08.2013

[URL-8]
http://www.oxfordjournals.org/nar/database/summary/603, verfügbar am 12.08.2013

[URL-9]
http://www.oxfordjournals.org/nar/database/summary/338, verfügbar am 12.08.2013

[URL-10]
http://de.wikipedia.org/wiki/VCL, verfügbar am 15.08.2013

[URL-11]
http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSM449176, verfügbar am 16.08.2013

[URL-12]
http://de.wikipedia.org/wiki/Object_Pascal, verfügbar am 16.08.2013

[URL-13]
http://media.affymetrix.com/support/technical/datasheets/human_datasheet.pdf, verfügbar am 18.08.2013

[URL-14]
http://rmaexpress.bmbolstad.com/, verfügbar am 18.08.2013

[URL-15]
http://affymetrix2.bioinf.fbb.msu.ru/probe_set.cgi?uniq_id=1575388, verfügbar am 20.08.2013

[URL-16]
http://www.delphi-treff.de/object-pascal/datentypen/#c4901, verfügbar am 05.09.2013

[URL-17]
http://de.wikipedia.org/wiki/Short_tandem_repeat, verfügbar am 11.09.2013

Anhang

Im Anhang sind Screenshots mit Beschreibungen der neuen, weiterentwickelten und verbesserten 'Differential-Expression'-Programmkomponente dargestellt. In dieser neuen Komponente wurden viele Funktionen integriert die erst aufgrund des Verwendens des anwenderfreundlicheren und schnelleren VirtualTreeview-Paketes in entsprechender Form umgesetzt werden konnten.

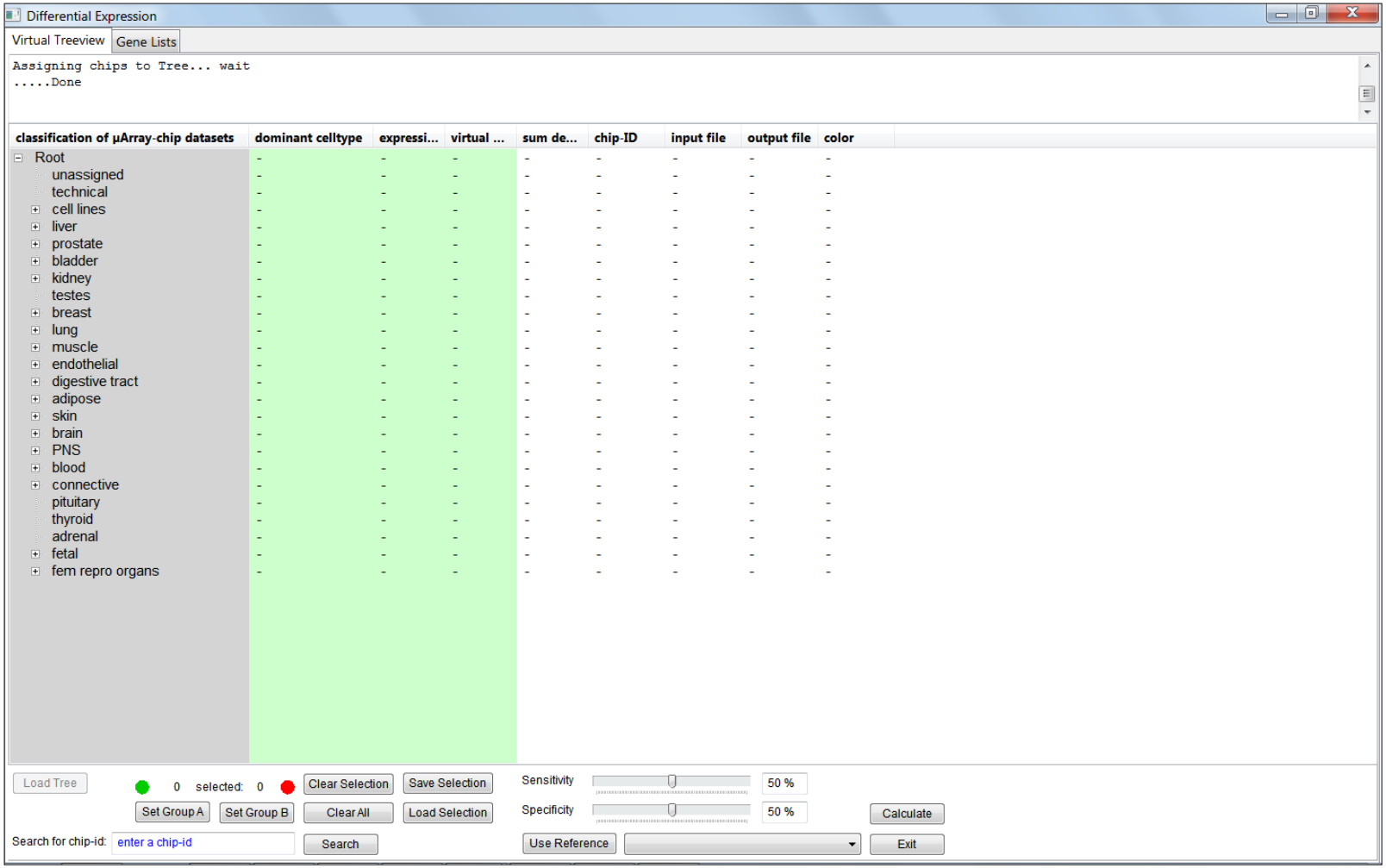


Abbildung 19 - Der VST nach der Initialisierung der Chip-Attribut
Nach dem Initialisieren aller Chip-Attribute wird vorerst nur der Root-Knoten angezeigt. Durch Erweitern des Root-Knotens zeigen sich die Knoten der ersten Ebene des Baums.

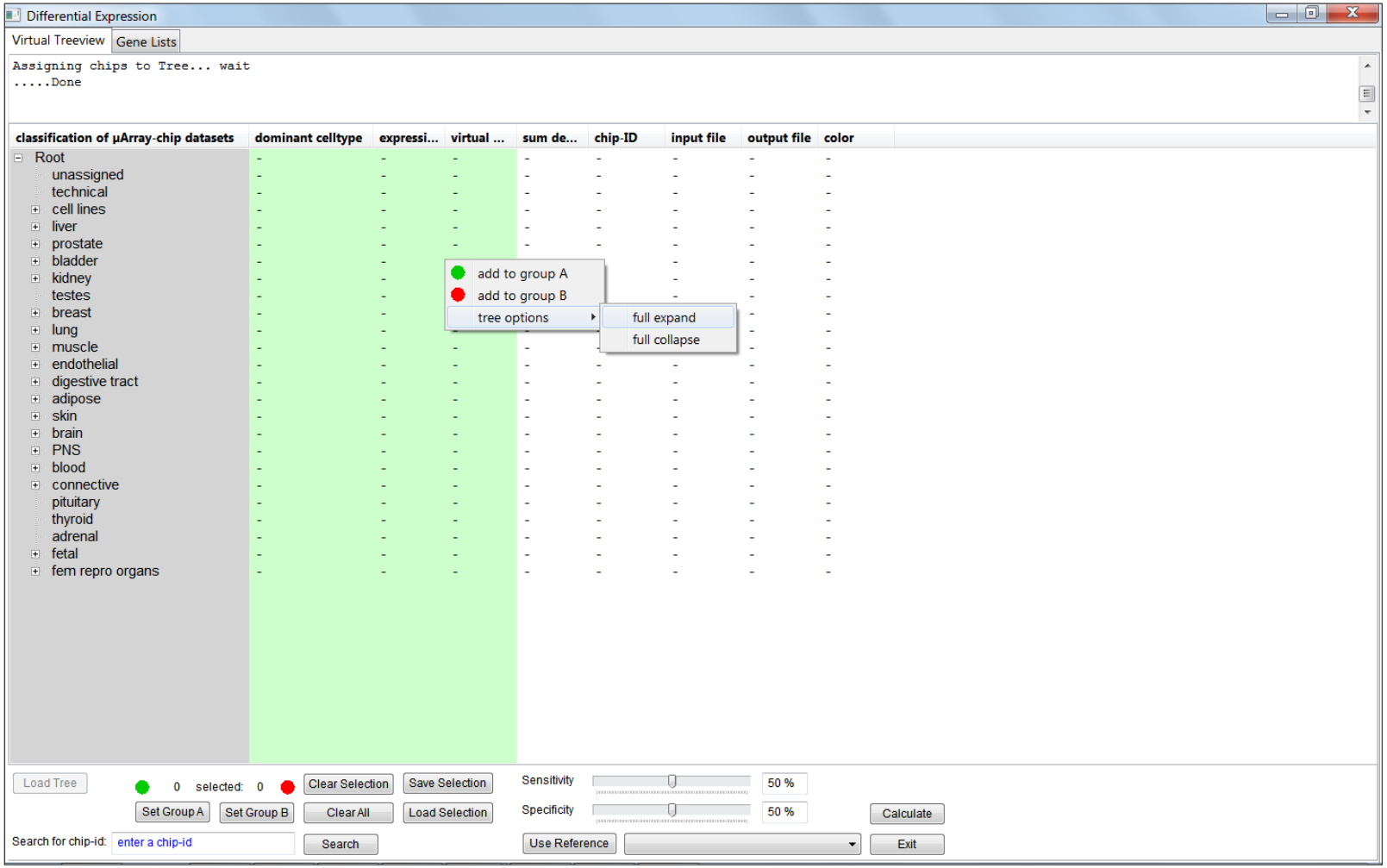


Abbildung 20 - Das Pop-up-Menü des VSTs

Das Pop-up-Menü besitzt neben den Einträgen zur Gruppierung der Knoten außerdem den Eintrag 'tree options', um alle Knoten komplett aus- oder einzuklappen.

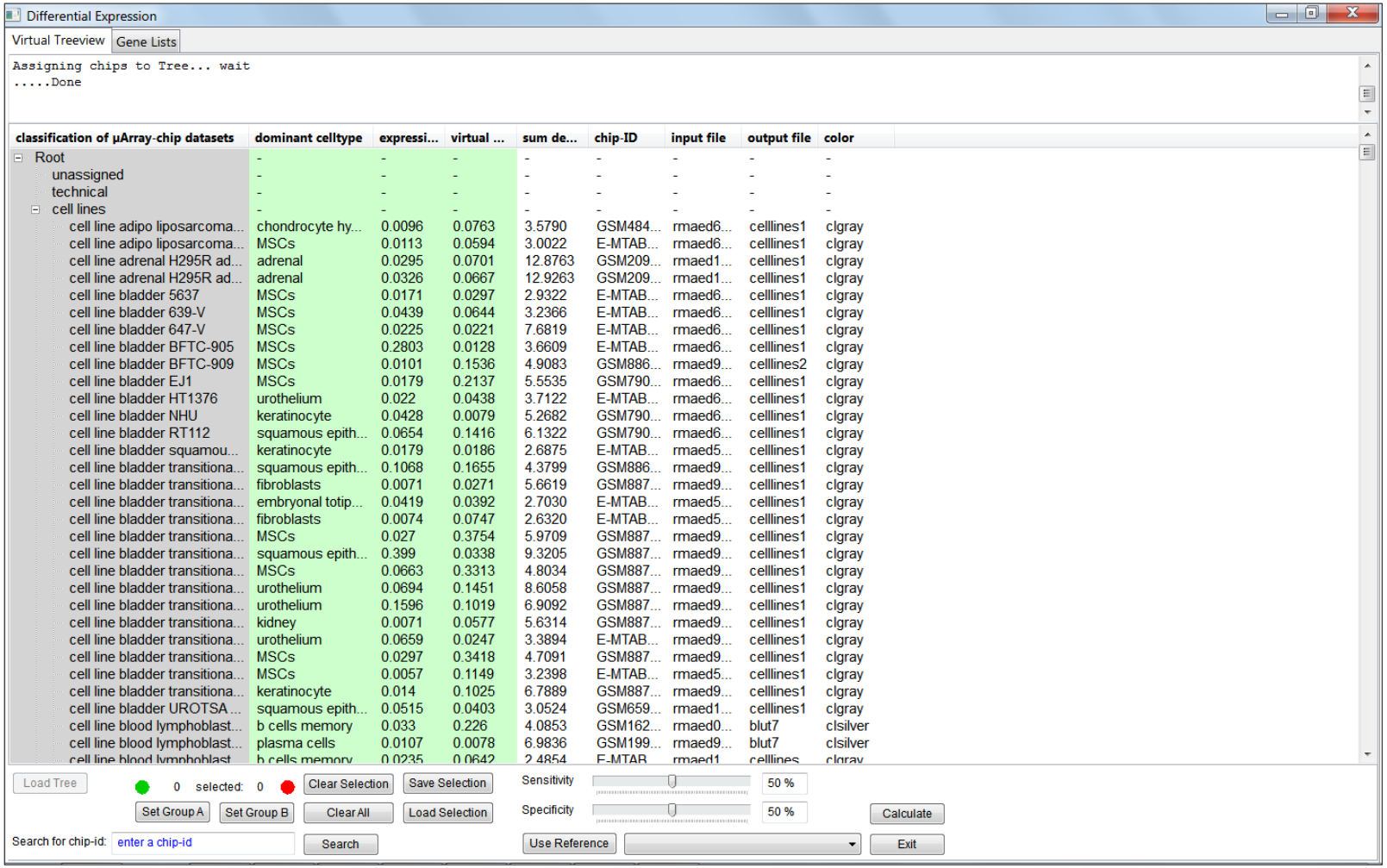


Abbildung 21 - Der komplett ausgeklappte VST

Zu sehen sind die 9 komplett gefüllten Spalten des VSTs. Die erste Spalte beinhaltet die Baumstruktur und die Chip-Beschreibung. Neu sind die grün und weiß hinterlegte Spalten, mit den biologischen sowie technischen Chip-Attributen, die als Auswahlkriterien dienen.

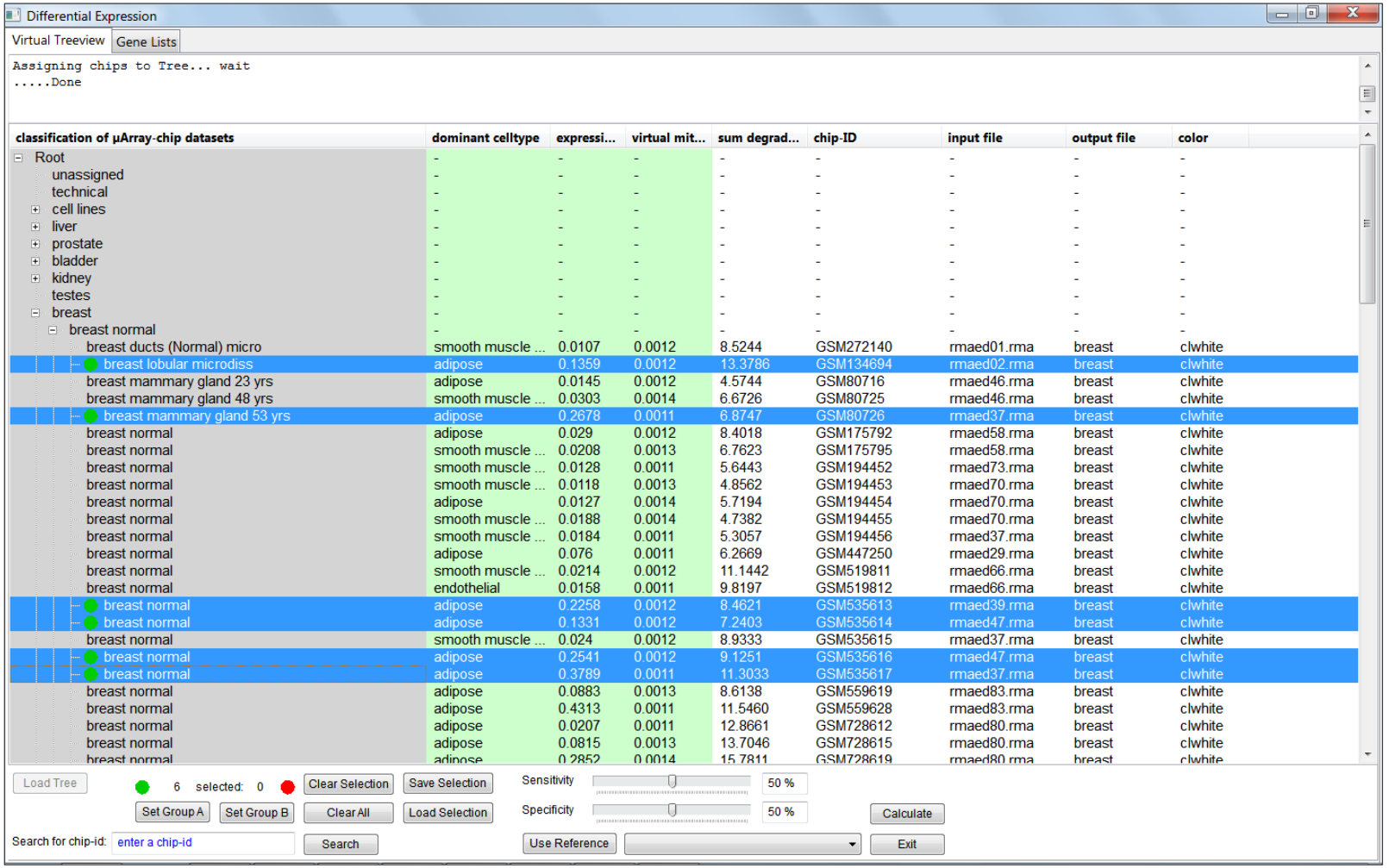


Abbildung 22 - Anwenderfreundliche Knotenauswahl im VST

Die Auswahl kann im VST unter Zuhilfenahme der Tasten SHIFT und STRG auf der Tastatur sowie durch das Aufziehen eines Auswahlrechtecks mit der Maus erfolgen. Hier wurden die ausgewählten Knoten schon zu Gruppe A hinzugefügt.

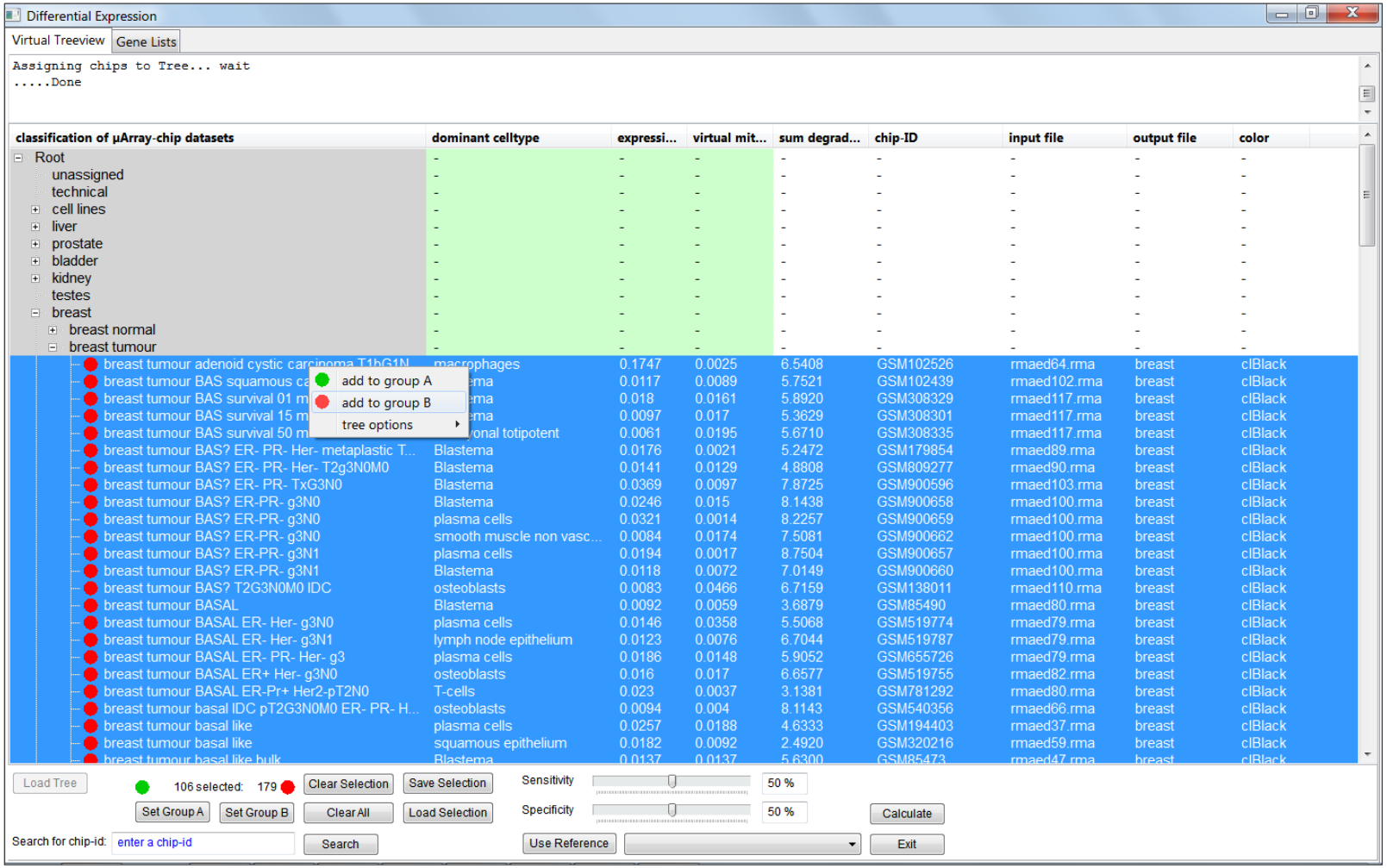


Abbildung 23 - Hinzufügen von Knoten zu einer Chip-Liste

Die Knoten können entweder durch den entsprechenden Eintrag im Pop-up-Menü oder durch Klicken auf den Button 'Set Group B' zur Gruppe B hinzugefügt werden. Dadurch werden die Knoten mit einem roten Punkt markiert.

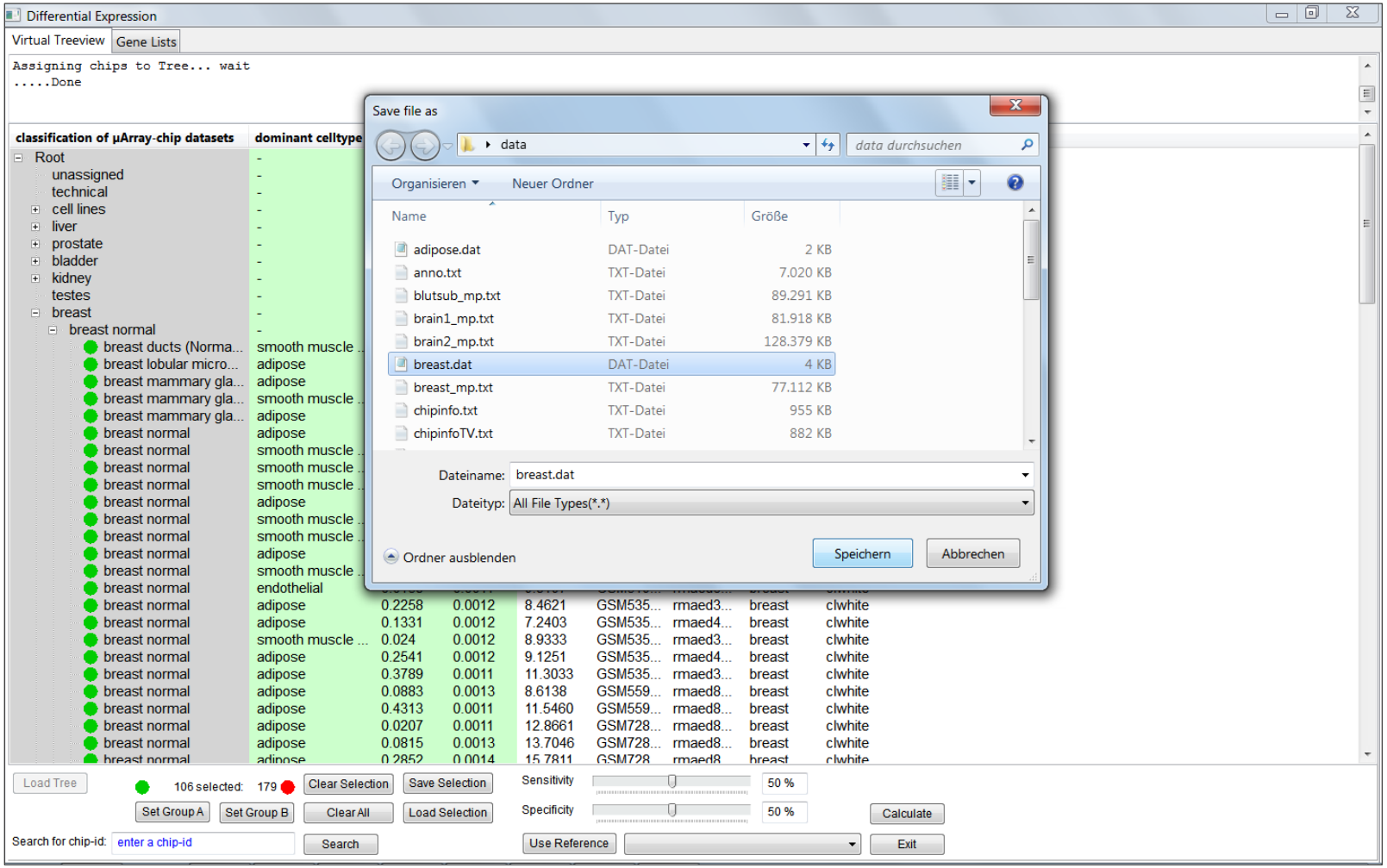


Abbildung 24 - Der Speicherdialog für die Chip-Listen

Eine Verbesserung der neuen 'Differential Expression'-Programmkomponente ist die Möglichkeit die Chip-Listen in Dateien abzuspeichern oder diese zu laden. Dazu wird nach Klicken auf den zugehörigen Button ein Speicher- bzw. Ladedialog geöffnet.

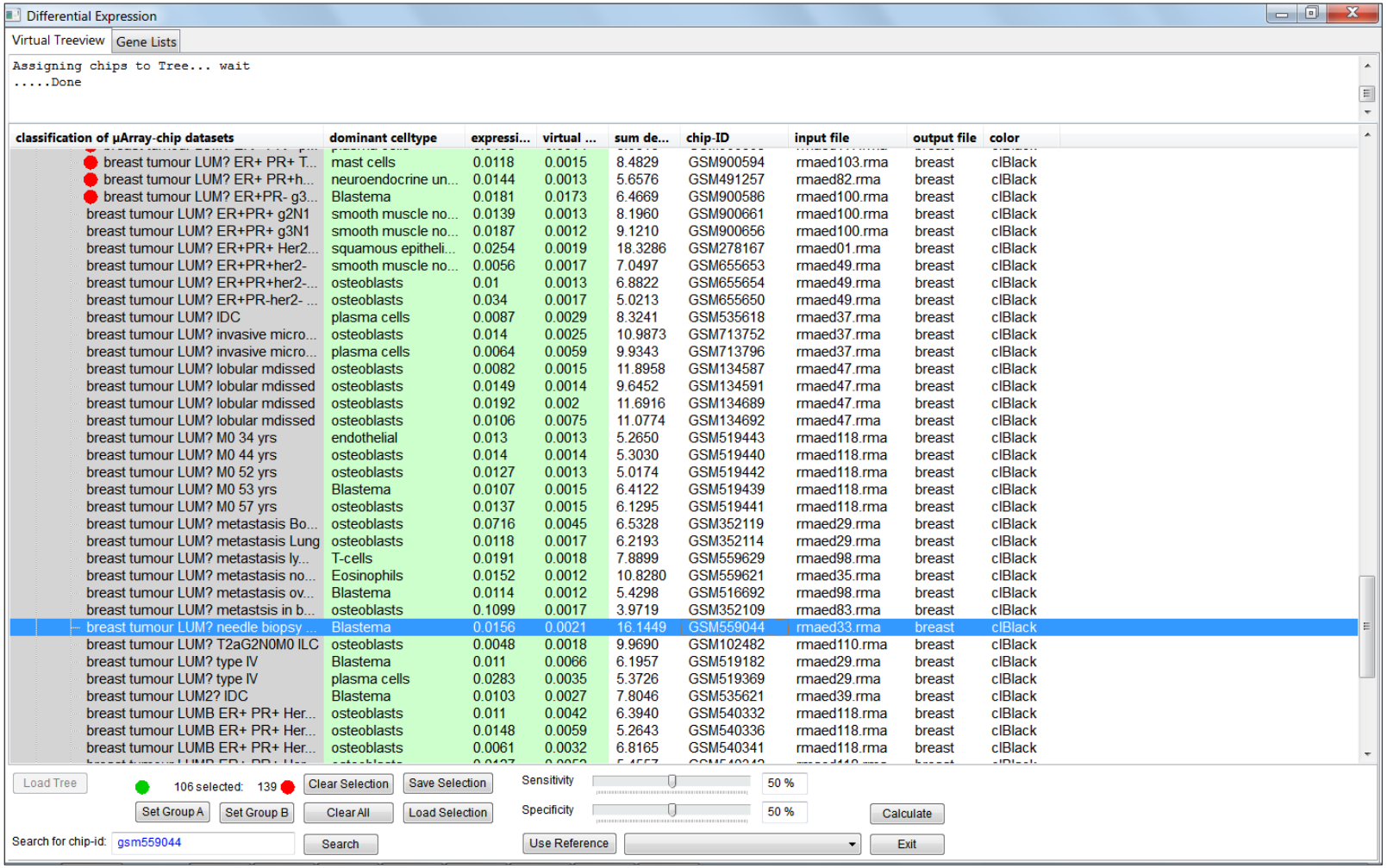


Abbildung 25 - Die Suche eines Eintrags im VST anhand einer Chip-ID

Eine wichtige Neuerung ist, dass nach Eingabe einer Chip-ID im Suchfeld und Klicken auf den Button 'Search' der Fokus des VSTs auf den zugehörige Knoten gerichtet und der richtige Eintrag markiert wird.

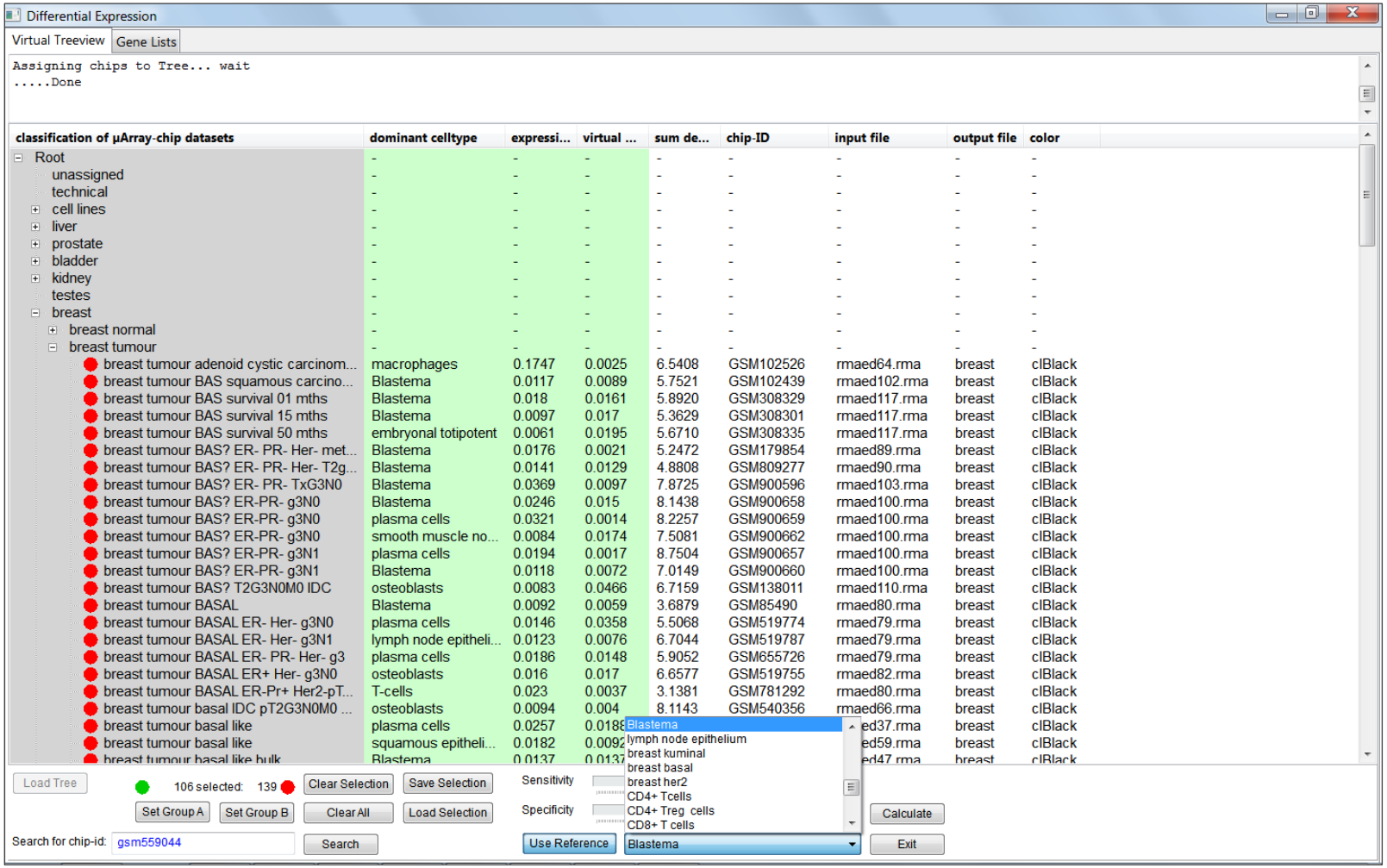


Abbildung 26 - Auswahl eines Zelltyps als Referenz

Die Möglichkeit einen Zelltyp als Referenz zu verwenden wurde aus der alten 'Differential Expression'-Komponente übernommen.

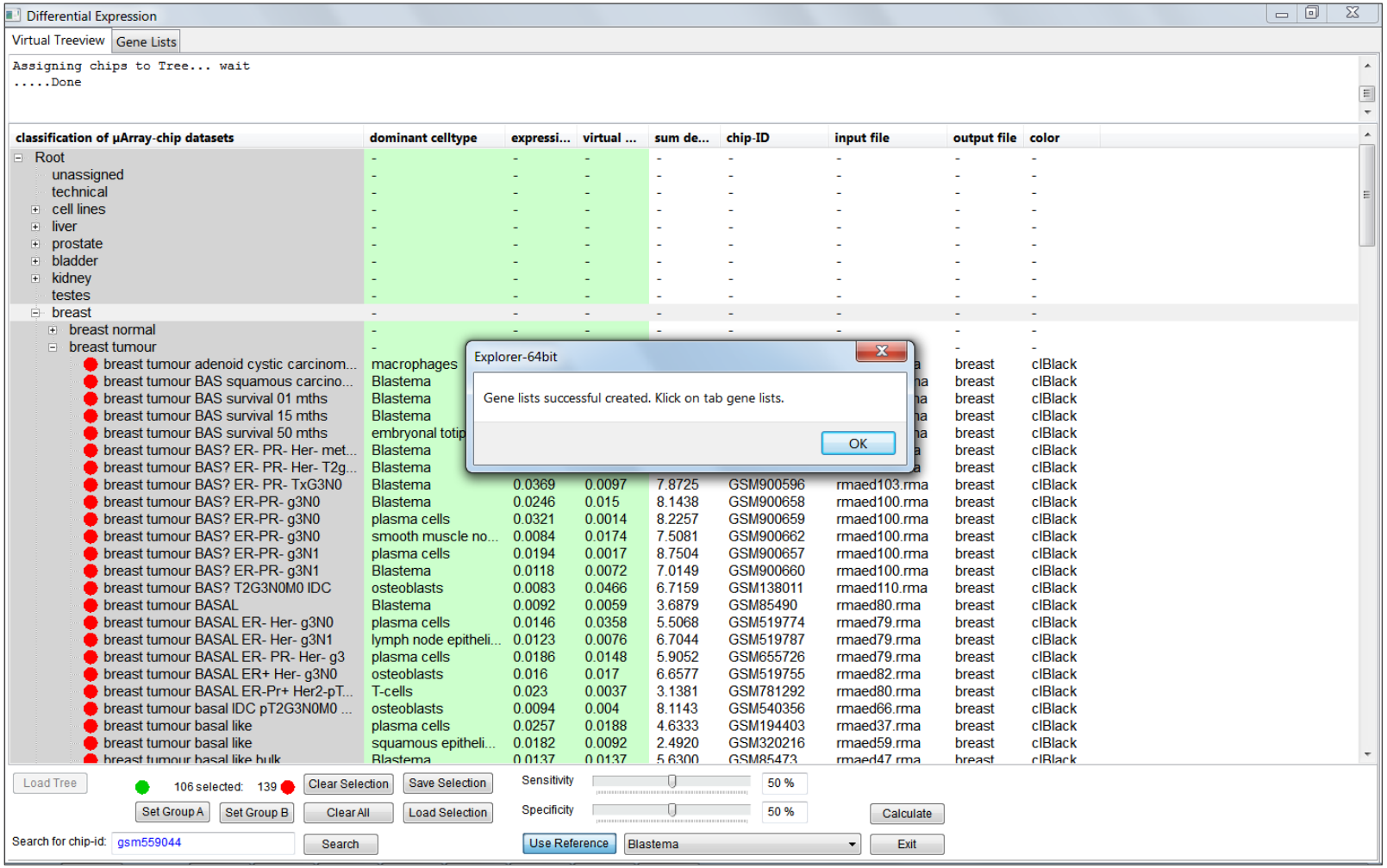
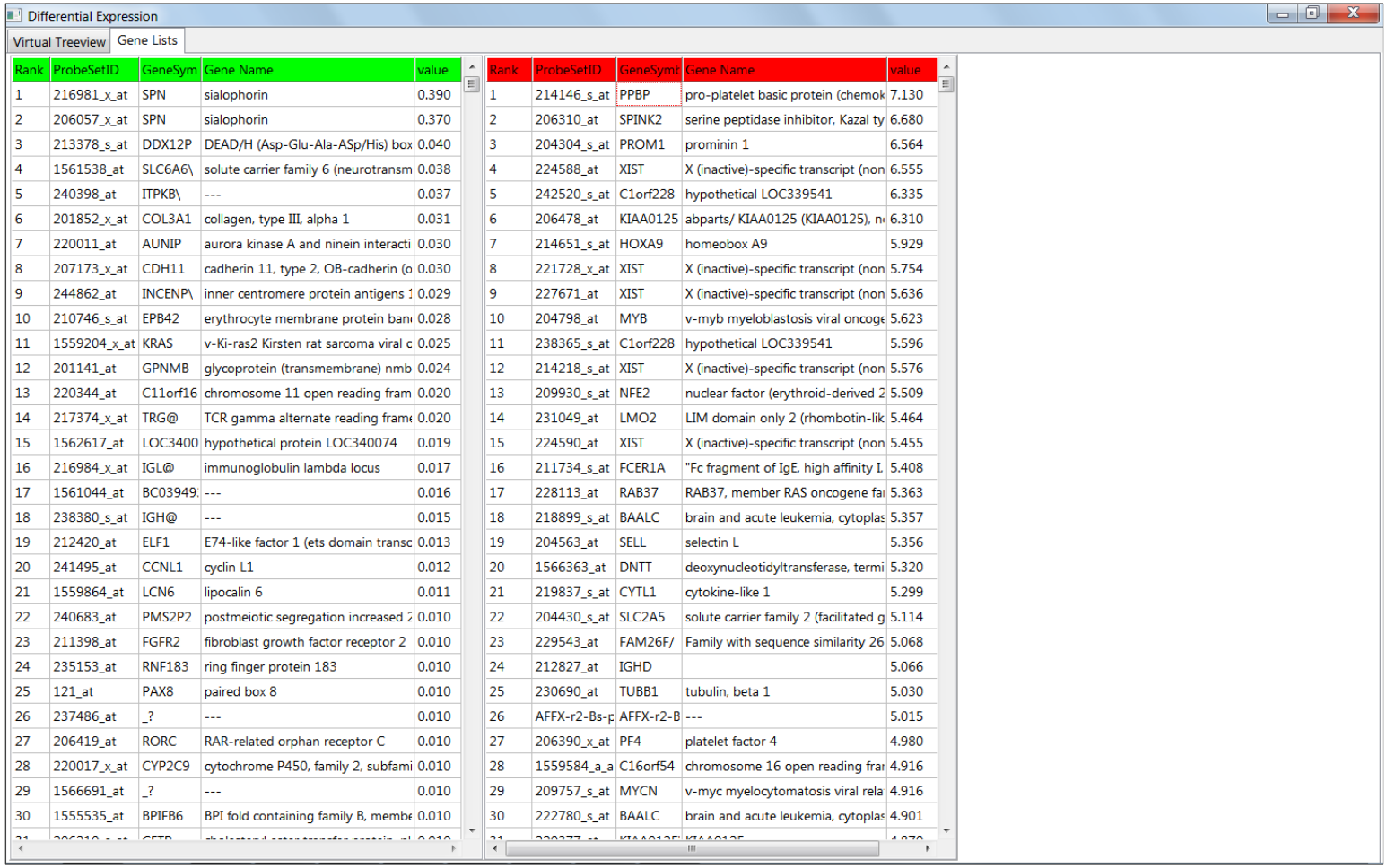


Abbildung 27 - Starten der Berechnung der Gen-Listen

Nach Klicken auf den Button 'Calculate' erfolgt die Berechnung der Gen-Listen unter Berücksichtigung aller zuvor festgelegten Parameter. Nach erfolgreicher Berechnung erscheint eine Informationsfenster.



Rank	ProbeSetID	GeneSym	Gene Name	value
1	216981_x_at	SPN	sialophorin	0.390
2	206057_x_at	SPN	sialophorin	0.370
3	213378_s_at	DDX12P	DEAD/H (Asp-Glu-Ala-ASp/His) box	0.040
4	1561538_at	SLC6A6\	solute carrier family 6 (neurotransm	0.038
5	240398_at	ITPKB\	---	0.037
6	201852_x_at	COL3A1	collagen, type III, alpha 1	0.031
7	220011_at	AUNIP	aurora kinase A and ninein interacti	0.030
8	207173_x_at	CDH11	cadherin 11, type 2, OB-cadherin (o	0.030
9	244862_at	INCENP\	inner centromere protein antigens 1	0.029
10	210746_s_at	EPB42	erythrocyte membrane protein ban	0.028
11	1559204_x_at	KRAS	v-Ki-ras2 Kirsten rat sarcoma viral c	0.025
12	201141_at	GPNMB	glycoprotein (transmembrane) nmb	0.024
13	220344_at	C11orf16	chromosome 11 open reading fram	0.020
14	217374_x_at	TRG@	TCR gamma alternate reading fram	0.020
15	1562617_at	LOC3400	hypothetical protein LOC340074	0.019
16	216984_x_at	IGL@	immunoglobulin lambda locus	0.017
17	1561044_at	BC03949	---	0.016
18	238380_s_at	IGH@	---	0.015
19	212420_at	ELF1	E74-like factor 1 (ets domain transc	0.013
20	241495_at	CCNL1	cyclin L1	0.012
21	1559864_at	LCN6	lipocalin 6	0.011
22	240683_at	PMS2P2	postmeiotic segregation increased 2	0.010
23	211398_at	FGFR2	fibroblast growth factor receptor 2	0.010
24	235153_at	RNF183	ring finger protein 183	0.010
25	121_at	PAX8	paired box 8	0.010
26	237486_at	_?	---	0.010
27	206419_at	RORC	RAR-related orphan receptor C	0.010
28	220017_x_at	CYP2C9	cytochrome P450, family 2, subfami	0.010
29	1566691_at	_?	---	0.010
30	1555535_at	BPIFB6	BPI fold containing family B, membe	0.010

Rank	ProbeSetID	GeneSymb	Gene Name	value
1	214146_s_at	PPBP	pro-platelet basic protein (chemok	7.130
2	206310_at	SPINK2	serine peptidase inhibitor, Kazal ty	6.680
3	204304_s_at	PROM1	prominin 1	6.564
4	224588_at	XIST	X (inactive)-specific transcript (non	6.555
5	242520_s_at	C1orf228	hypothetical LOC339541	6.335
6	206478_at	KIAA0125	abparts/ KIAA0125 (KIAA0125), n	6.310
7	214651_s_at	HOXA9	homeobox A9	5.929
8	221728_x_at	XIST	X (inactive)-specific transcript (non	5.754
9	227671_at	XIST	X (inactive)-specific transcript (non	5.636
10	204798_at	MYB	v-myb myeloblastosis viral oncoge	5.623
11	238365_s_at	C1orf228	hypothetical LOC339541	5.596
12	214218_s_at	XIST	X (inactive)-specific transcript (non	5.576
13	209930_s_at	NFE2	nuclear factor (erythroid-derived 2	5.509
14	231049_at	LMO2	LIM domain only 2 (rhombotin-lik	5.464
15	224590_at	XIST	X (inactive)-specific transcript (non	5.455
16	211734_s_at	FCER1A	"Fc fragment of IgE, high affinity I,	5.408
17	228113_at	RAB37	RAB37, member RAS oncogene fa	5.363
18	218899_s_at	BAALC	brain and acute leukemia, cytoplas	5.357
19	204563_at	SELL	selectin L	5.356
20	1566363_at	DNTT	deoxynucleotidyltransferase, termi	5.320
21	219837_s_at	CYTL1	cytokine-like 1	5.299
22	204430_s_at	SLC2A5	solute carrier family 2 (facilitated g	5.114
23	229543_at	FAM26F/	Family with sequence similarity 26	5.068
24	212827_at	IGHD		5.066
25	230690_at	TUBB1	tubulin, beta 1	5.030
26	AFFX-r2-Bs-p	AFFX-r2-B	---	5.015
27	206390_x_at	PF4	platelet factor 4	4.980
28	1559584_a_a	C16orf54	chromosome 16 open reading fra	4.916
29	209757_s_at	MYCN	v-myc myelocytomatosis viral rela	4.916
30	222780_s_at	BAALC	brain and acute leukemia, cytoplas	4.901

Abbildung 28 - Tabellen mit Gen-Listen

Die Gen-Listen werden aus den Chip-Listen erstellt. Die Tabellen beinhalten die ersten 1999 Probeset-Einträge mit den höchsten 'values'. Diese Gen-Listen sind das Ergebnis des Vergleichs der Expressionswerte aller Probesets mit den Chip-Gruppen A und B.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Mittweida, den 23.09.2013

Imre Katona